Research question:

Based on the data, which season should I come to (weather) or visit (championship) Lake Ko-Ko by comparing the winners of the championship (who to meet) and the weather (when to visit)? Based on this I can also predict each contestant each given category (such as relaxing, partying, etc.).

Resources

The Kimblers is from 10 years my data set what I've posting. This data contains all of the RBK games and data from 2002-2003. It contains the player's name, the game they won, and the date they won it.

Based on the data, I'll use the information to see if there's any trends in the game and determine which state can predict a championship winning team.

https://www.kaggle.com/datasets/mathurinache/the-kimblers/data?select=games_details.csv

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

import matplotlib.mathtext as mpt
import matplotlib.pyplot as plt
import numpy
import pandas
import sqlite3
from pandas.plotting import parallel_coordinates
```

Unique Function

The unique function filters out the 1 dates or no fail and returns one of each value in the list.

In [1]: DEF UNIQUE(LIST):
    UNIQUE = []
    FOR X IN LIST:
        IF X NOT IN UNIQUE:
            UNIQUE.APPEND(X)
    RETURN UNIQUE

Test UNIQUE FUNCTION

In [2]: LIST = [1, 2, 1, 3, 2, 4, 5]
PRINT("UNIQUE VALUES:", UNIQUE(LIST))

```
UNIQUE VALUES: [1, 2, 3, 4, 5]
```

In [3]: LIST1 = [0, 0, 0, 0, 0, 0, 0]
PRINT("UNIQUE VALUES:", UNIQUE(LIST1))

```
UNIQUE VALUES: [0]
```

This is one important and filtered by date. I removed multiple columns because they aren't needed for my results.

In [4]: DRIVER_GAME_PARQ_DATA = PD.READ_PARQUET("DRIVER_GAME.PARQUET")
DRIVER_GAME_PARQ_DATA.HEAD()

```
Out[4]:
   DRIVER_ID  GAME_ID  DRIVER_NAME  DRIVER_NUMBER  ...  DRIVER_DNF  DRIVER_DQ
0          1        1        NaN            NaN  ...       False      False
1          2        1        NaN            NaN  ...       False      False
2          3        1        NaN            NaN  ...       False      False
3          4        1        NaN            NaN  ...       False      False
4          5        1        NaN            NaN  ...       False      False

[5 rows x 10]
```

Seperating data into seasons

The first function is to filter out the data and seperate the seasons into the four kimbler's pitches. The four seasons are determined by the length of the data set by Tenth and Jacob's in the kimblers. Each season roughly has a 2000 player difference. (Player difference is how many players played in a season.)

In [5]: DEF SEASONSEPARATOR():
    GAMEDATA = PD.READ_PARQUET("GAME.PARQUET")
    GAMEDATA.HEAD()
    SEASON = []
    FOR X IN RANGE(GAMEDATA.SHAPE[0]):
        IF GAMEDATA.LOC[X, 'GAME_DATE'] == '2002-10-01':
            SEASON.APPEND('SPRING')
        ELIF GAMEDATA.LOC[X, 'GAME_DATE'] == '2002-12-01':
            SEASON.APPEND('SUMMER')
        ELIF GAMEDATA.LOC[X, 'GAME_DATE'] == '2003-02-01':
            SEASON.APPEND('FALL')
        ELIF GAMEDATA.LOC[X, 'GAME_DATE'] == '2003-04-01':
            SEASON.APPEND('WINTER')
        ELSE:
            SEASON.APPEND('NONE')
    GAMEDATA['SEASON'] = SEASON
    GAMEDATA.HEAD()
    GAMEDATA.TO_PARQUET('GAME.PARQUET')
    GAMEDATA.TO_CSV('GAME.CSV')
    PRINT('SEASONS SEPERATED')

TEST MAKING SEASONS

In [6]: PRINT(SEASONSEPARATOR())
    PRINT(PD.READ_PARQUET("GAME.PARQUET"))

```
SEASONS SEPERATED
   GAME_ID  ...  SEASON
0        1  ...  SPRING
1        2  ...  SPRING
2        3  ...  SPRING
3        4  ...  SPRING
4        5  ...  SPRING

[5 rows x 11]
```

GETID FUNCTION

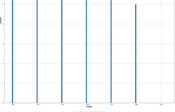
The function takes in the unique function to find all of the game IDs and add it to a specific list. The game ids is from TO Figure out which team played one and team 2 so I can use the average for each team and the grades player.

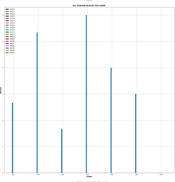
In [7]: DEF GETID(LIST):
    GAMEID = []
    FOR X IN LIST:
        IF X NOT IN GAMEID:
            GAMEID.APPEND(X)
    RETURN GAMEID
    PRINT("GAME IDS SEPERATED")

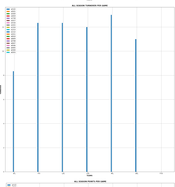
In [8]: GAMEID = GETID(LIST)
    PRINT(GAMEID)

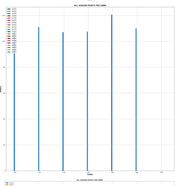
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41,
```

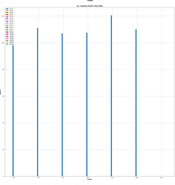
```
plt.ylabel('Frequency')
plt.xticks(rotation=45)
fig, ax = plt.subplots(1,1)
ax.bar(counts.index, counts, color=['red', 'red', 'red', '
```









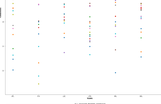


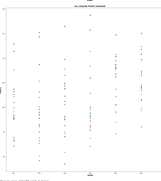
The function takes in data, figure regex, X axis, and the selected team to name. The function call out all of the functions above. We need the function because we will get a data dict in addition to having a header that we need, creating two dicts out of header.

```
In [1]: def dbgplot(data,fig_re,x_name=None):#takes in the data and header lines to index to plot respect. fig_re for the x label, and #name is the name of the header and axis data.
        ...
        # header separate
        header=data[0].split(sep=',')
        data=data[1:]
        #get axis values & functions into a lists#a function
        x=[int(d.split(sep=',')[0]) for d in data]
        y=[eval(d.split(sep=',')[1]) for d in data]
        ...
        #figure out data
        fig=int(header[1].split(sep=',')[0].replace(fig_re,''))
        ...
        # find workspace and index from 1 to numbers, assumed is in of problem
        if(x_name==None):x_name=header[0].replace(str(fig),'')
        ...
        plot.scatter(x,y,s=1,c=fig,cmap='viridis')
        plot.xlabel(x_name+str(fig))
        plot.ylabel('Value')
        plot.title('ALL '+header[1].replace(str(fig),'')+' PER GAME')
        ...
        return(x,y,fig)
```

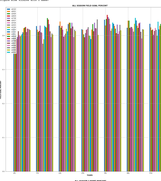
```
In [2]: dbgplot(data,'Chair',x_name='Game')
```

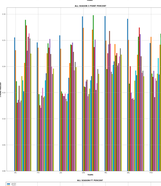
```
Out[2]:
(array([ 1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        
```

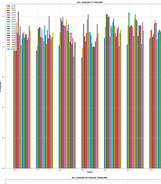


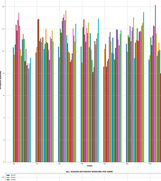


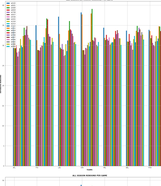

```
FFtgames season 2025/2026 game 1 Teams
```

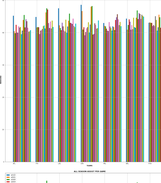


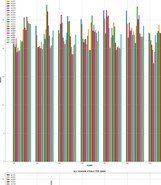


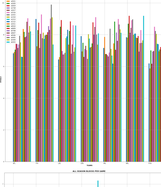


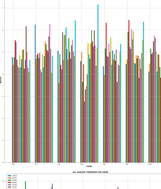


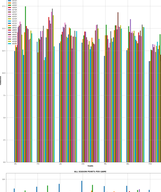


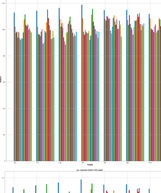




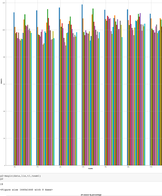


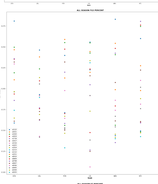


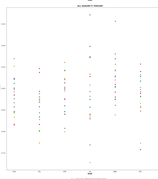


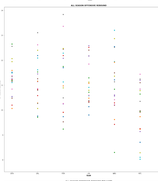



```
cd ~/NBA/
ffmpeg -r 1 -pattern_type glob -i "./plots/*.jpg" -c:v libx264 -r 30 -pix_fmt yuv420p -vf "scale=trunc(iw/2)*2:trunc(ih/2)*2" -y ./plots.mp4
cd
FFtgames season 2025/2026 game 1 Teams
```

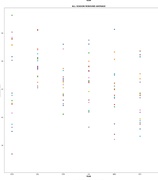


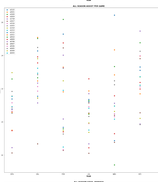

ALL SEASON TOS PERCENT
Year
Percentage


ALL SEASONS T PERCENT
Year
Percentage


ALL PERCENT OFFENSIVE REBOUND
Year
Percentage

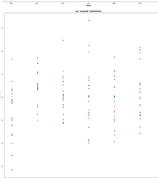
ALL SEASON OFFENSIVE REBOUND PER GAME
Year
Average

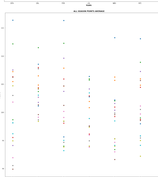

ALL SEASON REBOUND ASSISTS
Year
Average

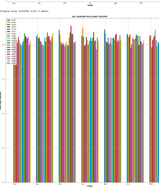

ALL SEASON BLOCK PER GAME
Year
Average

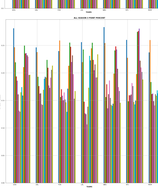
ALL SEASON STEAL ASSISTS
Year
Average

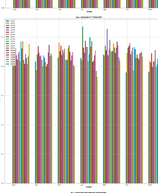
ALL SEASON BLOCK ASSISTS
Year
Average

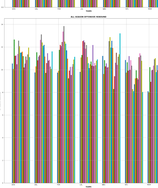

ALL SEASON TURNOVERS
Year
Average


ALL SEASON TURNOVER ASSISTS
Year
Average


Originals size 1920x1080 with 0 Issues

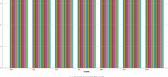

ALL SEASONS PERCENT FOUL PERCENT
Year
Percentage

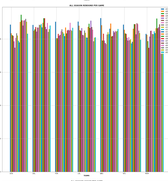

ALL SEASON FOUL PERCENT
Year
Percentage

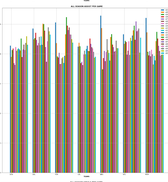

ALL SEASONS T PERCENT
Year
Percentage

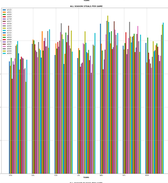
ALL SEASON DEFENSIVE REBOUND
Year
Average

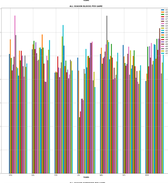
ALL SEASONS DEFENSIVE REBOUND PER GAME
Year
Average

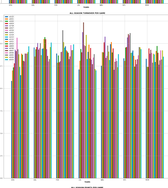


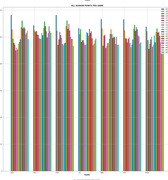


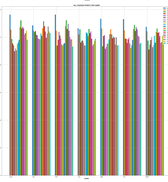




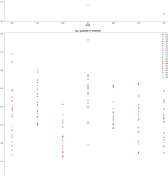


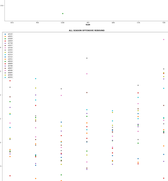






```
[1] 'ALL SEASON DEFICIT PER GAME'
[1] 'ALL SEASON SURPLUS PER GAME'
[1] 'ALL SEASON RAINFALL PER GAME'
[1] 'ALL SEASON DEFICIT PER GAME'
[1] 'ALL SEASON SURPLUS PER GAME'
[1] 'ALL SEASON RAINFALL PER GAME'
[1] 'ALL SEASON DEFICIT PER GAME'
[1] 'ALL SEASON SURPLUS PER GAME'
```

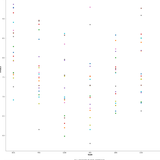


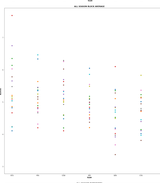


```
[1] 'ALL SEASON RAINFALL PER GAME'
[1] 'ALL SEASON DEFICIT PER GAME'
[1] 'ALL SEASON SURPLUS PER GAME'
[1] 'ALL SEASON RAINFALL PER GAME'
[1] 'ALL SEASON DEFICIT PER GAME'
[1] 'ALL SEASON SURPLUS PER GAME'
[1] 'ALL SEASON RAINFALL PER GAME'
[1] 'ALL SEASON DEFICIT PER GAME'
[1] 'ALL SEASON SURPLUS PER GAME'
```

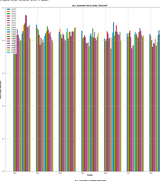


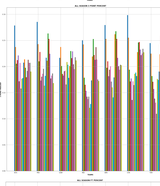
```
[1] 'ALL SEASON RAINFALL PER GAME'
[1] 'ALL SEASON DEFICIT PER GAME'
[1] 'ALL SEASON SURPLUS PER GAME'
[1] 'ALL SEASON RAINFALL PER GAME'
[1] 'ALL SEASON DEFICIT PER GAME'
[1] 'ALL SEASON SURPLUS PER GAME'
[1] 'ALL SEASON RAINFALL PER GAME'
[1] 'ALL SEASON DEFICIT PER GAME'
[1] 'ALL SEASON SURPLUS PER GAME'
```

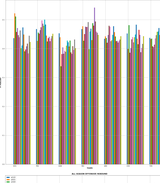


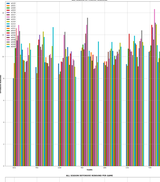


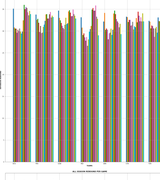
Progress plot [2025/05 2:24:00] 2025-05-01 0:00:00

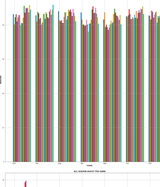


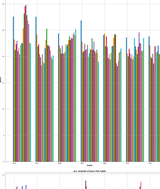


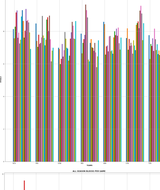


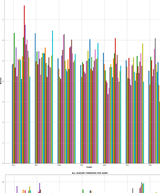


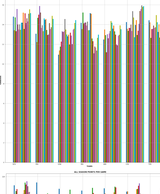


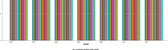


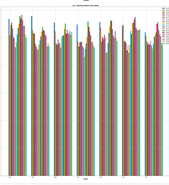










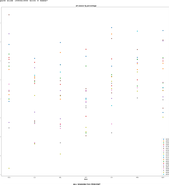


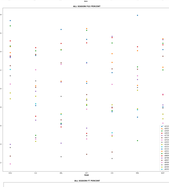

```
rm -rf
```

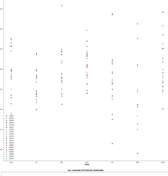
```
gsutil -m -o GSUTIL:parallel_thread_count=1 -o GSUTIL:parallel_process_count=1 -m -q -o "GSUTIL:boto_config=/home/runner/.config/gcloud/legacy_credentials/ci-service-account@firebase-ci-prod.iam.gserviceaccount.com/.boto" -o "GSUTIL:service_account=ci-service-account@firebase-ci-prod.iam.gserviceaccount.com" -o GSUTIL:parallel_composite_upload_threshold=150M -o GSUTIL:state_dir=/home/runner/.cache/gsutil -o GSUTIL:resumable_threshold=150M -o GSUTIL:rsync_buffer_lines=65536 -o GSUTIL:parallel_thread_count=24 -o GSUTIL:parallel_process_count=1 -c -d -e -r -x '^(\.git|.*\.(DS_Store|swp|tmp))$' ./ gs://firebase-ci-prod-ci-images
```

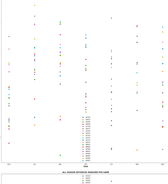
```
Copying file://./.DS_Store [Content-Type=application/octet-stream]
/-[1 files][  6.0 KiB/  6.0 KiB]   2 MiB/s
Operation completed over 1 objects/6.0 KiB.
```

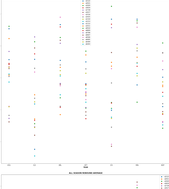
```
rm -rf
```

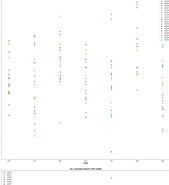


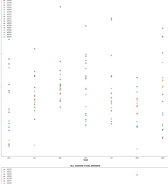


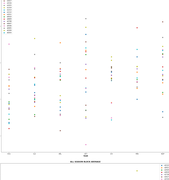


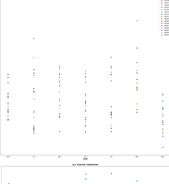


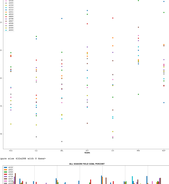




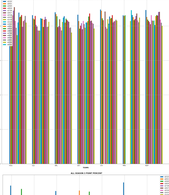


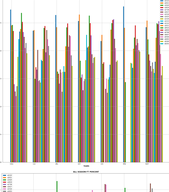


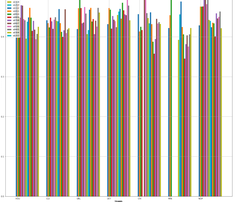


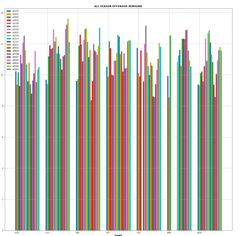


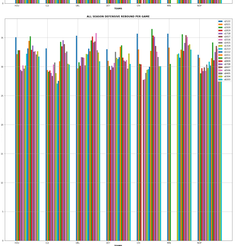

```
rm -rf
```

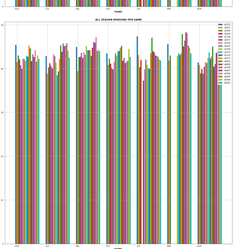


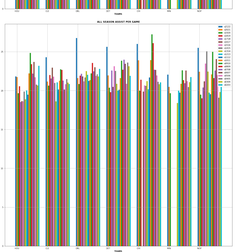


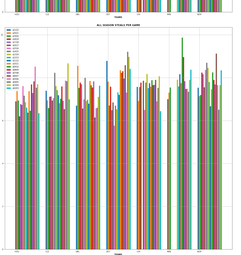


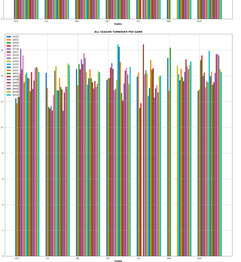


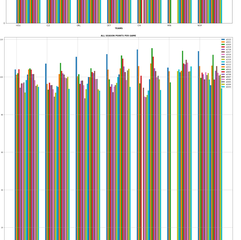












2021 champs are MIL. They are top in fg3, drebound, rebounds, points

1925 champs are LAL: non

1916 champs are TOR: non

1714 champs are DEN: They are top in orebound, steals, points

(note pts is top in alot of these categories)

1617 champs are DEN: They are top in fg3, steals,blk

1516 champs are CLE: non

1415 champs are DEN: They are top in rebounds

(alot of teams are top 5 in defensive rebounds)

1314 champs are SAN: They are top in non

1213 champs are MIA:They are top in non

1112 champs are MIA:They are top in drebound

1011 champs are CHI:, They are top in non

0910 champs are LAL: They are top in assist,steals,blks,

0809 champs are LAL:They are top in turnover

0708 champs are BOS:They are top in fg3

0607 champs are SAN:They are top in non

0506 champs are MIA:They are top in blk

0405 champs are SAN:They are top in non

0304 champs are DET:They are top in orebound rebounds

0203 champs are SAN:They are top in assist

0102 champs are LAL:They are top in reb,

Based on these graphs, it is 20% chance to a team to win a championship without being dominant in any category. There's a 65% chance for a team that is dominant in one category and its win rates, blocks, steals, and a 25% chance for a team that's dominate in the offensive side to win a championship(FG, FG3, FT, assist, points).

So the saying that defense wins games is kind of true, but then again PHX is top in a lot of these categories but their never won a championship in the last 20 years. Also notice that 35% of the time there is no correlation which can explain why the NBA finals is so exciting to watch.