NCAA Football Mileage Tracker

Summary

Due to the massive conference realignment that took place this offseason, college football teams are traveling more than ever before. I decided to take a look at the travel schedules of all 134 FBS teams, and determine which teams and conferences are traveling the most, as well as the least, for the upcoming 2024 season.

Import Libraries

In [571…

```
#Import all necessary libraries
import pandas as pd
import numpy as np
import geopandas as gp
import matplotlib.pyplot as plt
from shapely.geometry import Point
import missingno as msn
import seaborn as sns
import geopy
from geopy import distance
from geopy.distance import geodesic as dist
```

Stadium and Coordinates Data

Here, I read in the stadium data and coordinates. This file originally came from GitHub user gboeing as 'stadiums-geocoded.csv'. This dataset contains the stadium name and capacity, city, state, conference, latitude, and longitude for every FBS team. Note that when I read in the coordinates (which were supposed to be a tuple), they were reading in as a string; I simply deleted the 'coords' column and recreated it using the latitudes and longitudes listed for each team. The dataframe containing stadium data is shown below.

In [574…

```
#Read in stadium data/coordinates
df = pd.read_excel('/Users/kylechonko/Documents/Python/AllFBS.ods')
#Drop 'coords' since it was reading in as a string
df.drop(['coords'], axis = 1, inplace = True)
#Add coordinates again, this time as a tuple
df['coords'] = list(zip(df.latitude, df.longitude))
df
```

Out[574…

	team	conference	stadium	city	state	capacity	latitude	longitude	coords
0	Michigan	Big Ten	Michigan Stadium	Ann Arbor	MI	107601	42.265869	-83.748726	(42.2658687325174, -83.7487256526947)
1	Penn State	Big Ten	Beaver Stadium	University Park	PA	106572	40.812153	-77.856202	(40.8121527327504, -77.8562021255493)
2	Ohio State	Big Ten	Ohio Stadium	Columbus	OH	104944	40.001686	-83.019728	(40.0016856893694, -83.0197280645371)
3	Texas A&M	SEC	Kyle Field	College Station	TX	102733	30.610098	-96.340729	(30.6100975781748, -96.3407292285928)
4	Tennessee	SEC	Neyland Stadium	Knoxville	TN	102455	35.954734	-83.925333	(35.9547343726226, -83.9253330230713)
...	...	...	...	...	...	...	...	...	...
129	Liberty	C-USA	Williams Stadium	Lynchburg	VA	19200	37.354414	-79.175047	(37.354414, -79.175047)
130	Sam Houston State	C-USA	Bowers Stadium	Huntsville	TX	14000	30.713947	-95.541916	(30.7139467376334, -95.5419158935547)
131	Coastal Carolina	Sun Belt	Brooks Stadium	Conway	SC	9214	33.792838	-79.017023	(33.7928381752245, -79.0170228280526)
132	Kennesaw State	C-USA	Fifth Third Bank Stadium	Kennesaw	GA	8318	34.023434	-84.615490	(34.0234337, -84.6154897)
133	UAB	American	Protective Stadium	Birmingham	AL	47100	33.527800	-86.809200	(33.5278, -86.8092)

134 rows × 9 columns

Schedule Data

Next, I read in the schedule data for each team. This data came from FBSchedules.com, and was initially read into an Excel workbook. The location for every game was listed for the corresponding teams, from weeks 0-15. For example, Akron's week 1 game was listed as

'Ohio State'. However, Ohio State's did not say 'Akron', but rather 'Ohio State', since the game was played on Ohio State's campus. For neutral site games, I listed the FBS stadium that is closest in proximity to the site where the game was played. So, Clemson vs. Georgia in Mercedes-Benz Stadium in Atlanta was listed as 'Georgia Tech' for both teams. For bye weeks, cells were left blank.

In [577…
```
#Read in Schedule Data from NCAASchedule2024 file
schedule = pd.read_excel('/Users/kylechonko/Documents/Python/NCAASchedule2024.ods')
#Drop and re-add coords to dataframe, since again reading in as a string
schedule.drop(['coords'], axis = 1, inplace = True)
schedule['coords'] = list(zip(df.latitude, df.longitude))
#Reorder schedule dataframe
schedule = schedule[['team','conference', 'latitude','longitude','coords','Wk0','Wk1','Wk2','Wk3','Wk4','Wk5','
schedule
```

Out[577…

	team	conference	latitude	longitude	coords	Wk0	Wk1	Wk2	Wk3	Wk4	...	
0	Michigan	Big Ten	42.265869	-83.748726	(42.2658687325174, -83.7487256526947)	NaN	Michigan	Michigan	Michigan	Michigan	...	Wa
1	Penn State	Big Ten	40.812153	-77.856202	(40.8121527327504, -77.8562021255493)	NaN	West Virginia	Penn State	NaN	Penn State	...	Pe
2	Ohio State	Big Ten	40.001686	-83.019728	(40.0016856893694, -83.0197280645371)	NaN	Ohio State	Ohio State	NaN	Ohio State	...	Ol
3	Texas A&M	SEC	30.610098	-96.340729	(30.6100975781748, -96.3407292285928)	NaN	Texas A&M	Texas A&M	Florida	Texas A&M	...	Te
4	Tennessee	SEC	35.954734	-83.925333	(35.9547343726226, -83.9253330230713)	NaN	Tennessee	Charlotte	Tennessee	Oklahoma	...	A
...	...	...	...	...	...	...	...	...	...	...	...	
129	Liberty	C-USA	37.354414	-79.175047	(37.354414, -79.175047)	NaN	Liberty	New Mexico State	Liberty	Liberty	...	
130	Sam Houston State	C-USA	30.713947	-95.541916	(30.7139467376334, -95.5419158935547)	NaN	Rice	UCF	Sam Houston State	Sam Houston State	...	
131	Coastal Carolina	Sun Belt	33.792838	-79.017023	(33.7928381752245, -79.0170228280526)	NaN	Jacksonville State	Coastal Carolina	Temple	Coastal Carolina	...	
132	Kennesaw State	C-USA	34.023434	-84.615490	(34.0234337, -84.6154897)	NaN	UTSA	Kennesaw State	San Jose State	NaN	...	K
133	UAB	American	33.527800	-86.809200	(33.5278, -86.8092)	NaN	UAB	Louisiana-Monroe	Arkansas	NaN	...	

134 rows × 21 columns

schedulechanger function

Next, I created a function to change all null values in a row to the team name, since teams would not travel for a bye week. In the original file, any bye weeks were left as a blank cell. When read into Python, they became NaN values. Here, I changed these values to the name of the team listed in each row, since later on the distance calculator will read these as 0.

In [580…
```
def schedulechanger(df, col, team):
    for i in df.index:
        val = df[col].iloc[i]
        if pd.isna(val):
            df.at[i,col] = df.team[i]
        else:
            ()
    return df

schedulechanger(schedule, 'Wk0', 'team')
schedulechanger(schedule, 'Wk1', 'team')
schedulechanger(schedule, 'Wk2', 'team')
schedulechanger(schedule, 'Wk3', 'team')
schedulechanger(schedule, 'Wk4', 'team')
schedulechanger(schedule, 'Wk5', 'team')
schedulechanger(schedule, 'Wk6', 'team')
schedulechanger(schedule, 'Wk7', 'team')
schedulechanger(schedule, 'Wk8', 'team')
schedulechanger(schedule, 'Wk9', 'team')
schedulechanger(schedule, 'Wk10', 'team')
schedulechanger(schedule, 'Wk11', 'team')
schedulechanger(schedule, 'Wk12', 'team')
schedulechanger(schedule, 'Wk13', 'team')
schedulechanger(schedule, 'Wk14', 'team')
schedulechanger(schedule, 'Wk15', 'team')
```

Out[580…

	team	conference	latitude	longitude	coords	Wk0	Wk1	Wk2	Wk3	Wk4	.
0	Michigan	Big Ten	42.265869	-83.748726	(42.2658687325174, -83.7487256526947)	Michigan	Michigan	Michigan	Michigan	Michigan	.
1	Penn State	Big Ten	40.812153	-77.856202	(40.8121527327504, -77.8562021255493)	Penn State	West Virginia	Penn State	Penn State	Penn State	.
2	Ohio State	Big Ten	40.001686	-83.019728	(40.0016856893694, -83.0197280645371)	Ohio State	Ohio State	Ohio State	Ohio State	Ohio State	.
3	Texas A&M	SEC	30.610098	-96.340729	(30.6100975781748, -96.3407292285928)	Texas A&M	Texas A&M	Texas A&M	Florida	Texas A&M	.
4	Tennessee	SEC	35.954734	-83.925333	(35.9547343726226, -83.9253330230713)	Tennessee	Tennessee	Charlotte	Tennessee	Oklahoma	.
...	...	...	...	...	...	...	...	...	...	...	.
129	Liberty	C-USA	37.354414	-79.175047	(37.354414, -79.175047)	Liberty	Liberty	New Mexico State	Liberty	Liberty	.
130	Sam Houston State	C-USA	30.713947	-95.541916	(30.7139467376334, -95.5419158935547)	Sam Houston State	Rice	UCF	Sam Houston State	Sam Houston State	.
131	Coastal Carolina	Sun Belt	33.792838	-79.017023	(33.7928381752245, -79.0170228280526)	Coastal Carolina	Jacksonville State	Coastal Carolina	Temple	Coastal Carolina	.
132	Kennesaw State	C-USA	34.023434	-84.615490	(34.0234337, -84.6154897)	Kennesaw State	UTSA	Kennesaw State	San Jose State	Kennesaw State	.
133	UAB	American	33.527800	-86.809200	(33.5278, -86.8092)	UAB	UAB	Louisiana-Monroe	Arkansas	UAB	.

134 rows × 21 columns

Latitude and Longitude dictionaries

Next, I created dictionaries that contained the latitudes and longitudes for each school. The idea here was to swap out the school names for the corresponding latitudes and longitudes, so I could use the geopy.distance function to calculate distances with ease.

In [583…

```
coords_dict = dict(zip(df.team, df.coords))
lat_dict = dict(zip(df.team, df.latitude))
long_dict = dict(zip(df.team, df.longitude))
```

Breaking up the school names

Originally, I wanted to swap the teams listed on the schedule for their corresponding coordinates listed within the coords_dict dictionary shown above. However, later on when I tried to use the geopy.distance function on these coordinates, I received an error message. I decided that the best course of action was to break each week into a latitude and longitude column, read in the coordinates from the lat and long dicts, and then join these data points back together later on when the geopy.distance function would be utilized. Each column containing schedule data was duplicated and renamed (ex:'Wk1' was replaced with 'Wk1lat' and 'Wk1long'. The code below shows me breaking down each week into lat and long.

In [586…

```
#Break up school name into latitude and longitude (so we can swap them out with the dictionaries we created ear
#Here, I just duplicated each column for weeks 0-15
schedule = schedule[['team','conference','latitude','longitude','coords','Wk0','Wk0','Wk1','Wk1','Wk2','Wk2','W
#Rename columns (to eliminate duplicate column names and to break up into WkXXlat, WkXXlong
schedule.columns = ['team','conference','latitude','longitude','coords','Wk0lat','Wk0long','Wk1lat','Wk1long','
schedule
```

Out[586…

	team	conference	latitude	longitude	coords	Wk0lat	Wk0long	Wk1lat	Wk1long	Wk2lat
0	Michigan	Big Ten	42.265869	-83.748726	(42.2658687325174, -83.7487256526947)	Michigan	Michigan	Michigan	Michigan	Michigan
1	Penn State	Big Ten	40.812153	-77.856202	(40.8121527327504, -77.8562021255493)	Penn State	Penn State	West Virginia	West Virginia	Penn State
2	Ohio State	Big Ten	40.001686	-83.019728	(40.0016856893694, -83.0197280645371)	Ohio State	Ohio State	Ohio State	Ohio State	Ohio State
3	Texas A&M	SEC	30.610098	-96.340729	(30.6100975781748, -96.3407292285928)	Texas A&M	Texas A&M	Texas A&M	Texas A&M	Texas A&M
4	Tennessee	SEC	35.954734	-83.925333	(35.9547343726226, -83.9253330230713)	Tennessee	Tennessee	Tennessee	Tennessee	Charlotte
...	...	...	...	...	...	...	...	...	...	...
129	Liberty	C-USA	37.354414	-79.175047	(37.354414, -79.175047)	Liberty	Liberty	Liberty	Liberty	New Mexico State
130	Sam Houston State	C-USA	30.713947	-95.541916	(30.7139467376334, -95.5419158935547)	Sam Houston State	Sam Houston State	Rice	Rice	UCF
131	Coastal Carolina	Sun Belt	33.792838	-79.017023	(33.7928381752245, -79.0170228280526)	Coastal Carolina	Coastal Carolina	Jacksonville State	Jacksonville State	Coastal Carolina
132	Kennesaw State	C-USA	34.023434	-84.615490	(34.0234337, -84.6154897)	Kennesaw State	Kennesaw State	UTSA	UTSA	Kennesaw State
133	UAB	American	33.527800	-86.809200	(33.5278, -86.8092)	UAB	UAB	UAB	UAB	Louisiana-Monroe

134 rows × 37 columns

Swapping out the school names for their latitudes and longitudes

Here, I utilized the lat and long dicts I created earlier, and replaced the school names with the latitudes and longitudes of the school listed for each week.

In [589…

```
#Turn School Names into Latitude and Longitude
schedule['Wk0lat'].replace(lat_dict, inplace = True)
schedule['Wk0long'].replace(long_dict, inplace =True)
schedule['Wk1lat'].replace(lat_dict, inplace = True)
schedule['Wk1long'].replace(long_dict, inplace =True)
schedule['Wk2lat'].replace(lat_dict, inplace = True)
schedule['Wk2long'].replace(long_dict, inplace =True)
schedule['Wk3lat'].replace(lat_dict, inplace = True)
schedule['Wk3long'].replace(long_dict, inplace =True)
schedule['Wk4lat'].replace(lat_dict, inplace = True)
schedule['Wk4long'].replace(long_dict, inplace =True)
schedule['Wk5lat'].replace(lat_dict, inplace = True)
schedule['Wk5long'].replace(long_dict, inplace =True)
schedule['Wk6lat'].replace(lat_dict, inplace = True)
schedule['Wk6long'].replace(long_dict, inplace =True)
schedule['Wk7lat'].replace(lat_dict, inplace = True)
schedule['Wk7long'].replace(long_dict, inplace =True)
schedule['Wk8lat'].replace(lat_dict, inplace = True)
schedule['Wk8long'].replace(long_dict, inplace =True)
schedule['Wk9lat'].replace(lat_dict, inplace = True)
schedule['Wk9long'].replace(long_dict, inplace =True)
schedule['Wk10lat'].replace(lat_dict, inplace = True)
schedule['Wk10long'].replace(long_dict, inplace =True)
schedule['Wk11lat'].replace(lat_dict, inplace = True)
schedule['Wk11long'].replace(long_dict, inplace =True)
schedule['Wk12lat'].replace(lat_dict, inplace = True)
schedule['Wk12long'].replace(long_dict, inplace =True)
schedule['Wk13lat'].replace(lat_dict, inplace = True)
schedule['Wk13long'].replace(long_dict, inplace =True)
schedule['Wk14lat'].replace(lat_dict, inplace = True)
schedule['Wk14long'].replace(long_dict, inplace =True)
schedule['Wk15lat'].replace(lat_dict, inplace = True)
schedule['Wk15long'].replace(long_dict, inplace =True)
schedule
```

```
/var/folders/6p/v9rm9dqj4xq8113by2w3q15m0000gn/T/ipykernel_20020/2322106280.py:2: FutureWarning: A value is tryi
ng to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on w
hich we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)'
```

```
rning-a-view-versus-a-copy
  schedule['Wk15long'].replace(long_dict, inplace =True)
```

Out[589…

	team	conference	latitude	longitude	coords	Wk0lat	Wk0long	Wk1lat	Wk1long	Wk2lat	..
0	Michigan	Big Ten	42.265869	-83.748726	(42.2658687325174, -83.7487256526947)	42.265869	-83.748726	42.265869	-83.748726	42.265869	..
1	Penn State	Big Ten	40.812153	-77.856202	(40.8121527327504, -77.8562021255493)	40.812153	-77.856202	39.652220	-79.955175	40.812153	..
2	Ohio State	Big Ten	40.001686	-83.019728	(40.0016856893694, -83.0197280645371)	40.001686	-83.019728	40.001686	-83.019728	40.001686	..
3	Texas A&M	SEC	30.610098	-96.340729	(30.6100975781748, -96.3407292285928)	30.610098	-96.340729	30.610098	-96.340729	30.610098	..
4	Tennessee	SEC	35.954734	-83.925333	(35.9547343726226, -83.9253330230713)	35.954734	-83.925333	35.954734	-83.925333	35.310281	..
...	...	...	...	...	...	...	...	...	...	...	..
129	Liberty	C-USA	37.354414	-79.175047	(37.354414, -79.175047)	37.354414	-79.175047	37.354414	-79.175047	32.279749	..
130	Sam Houston State	C-USA	30.713947	-95.541916	(30.7139467376334, -95.5419158935547)	30.713947	-95.541916	29.716234	-95.409265	28.607959	..
131	Coastal Carolina	Sun Belt	33.792838	-79.017023	(33.7928381752245, -79.0170228280526)	33.792838	-79.017023	33.820310	-85.766466	33.792838	..
132	Kennesaw State	C-USA	34.023434	-84.615490	(34.0234337, -84.6154897)	34.023434	-84.615490	29.416937	-98.478903	34.023434	..
133	UAB	American	33.527800	-86.809200	(33.5278, -86.8092)	33.527800	-86.809200	33.527800	-86.809200	32.531069	..

134 rows × 37 columns

Zipping the lats and longs

I then zipped the lats and longs for each stadium into a tuple, and created a new column within the dataframe that contained this tuple for each week of the season. I then dropped the 'WkXXlat' and 'WkXXlong' columns from the dataframe since they were no longer necessary.

In [592…

```
#Zip the lats and longs for each stadium into a tuple for each week
schedule['wk0'] = list(zip(schedule.Wk0lat, schedule.Wk0long))
schedule['wk1'] = list(zip(schedule.Wk1lat, schedule.Wk1long))
schedule['wk2'] = list(zip(schedule.Wk2lat, schedule.Wk2long))
schedule['wk3'] = list(zip(schedule.Wk3lat, schedule.Wk3long))
schedule['wk4'] = list(zip(schedule.Wk4lat, schedule.Wk4long))
schedule['wk5'] = list(zip(schedule.Wk5lat, schedule.Wk5long))
schedule['wk6'] = list(zip(schedule.Wk6lat, schedule.Wk6long))
schedule['wk7'] = list(zip(schedule.Wk7lat, schedule.Wk7long))
schedule['wk8'] = list(zip(schedule.Wk8lat, schedule.Wk8long))
schedule['wk9'] = list(zip(schedule.Wk9lat, schedule.Wk9long))
schedule['wk10'] = list(zip(schedule.Wk10lat, schedule.Wk10long))
schedule['wk11'] = list(zip(schedule.Wk11lat, schedule.Wk11long))
schedule['wk12'] = list(zip(schedule.Wk12lat, schedule.Wk12long))
schedule['wk13'] = list(zip(schedule.Wk13lat, schedule.Wk13long))
schedule['wk14'] = list(zip(schedule.Wk14lat, schedule.Wk14long))
schedule['wk15'] = list(zip(schedule.Wk15lat, schedule.Wk15long))
#Drop the 'wkXXlat' and 'wkXXlong' from the dataframe
schedule = schedule[['team', 'conference','coords','wk0','wk1','wk2','wk3','wk4','wk5','wk6','wk7','wk8','wk9',
```

```
/var/folders/6p/v9rm9dqj4xq8113by2w3q15m0000gn/T/ipykernel_20020/3267419667.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#retu
rning-a-view-versus-a-copy
  schedule['wk0'] = list(zip(schedule.Wk0lat, schedule.Wk0long))
/var/folders/6p/v9rm9dqj4xq8113by2w3q15m0000gn/T/ipykernel_20020/3267419667.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#retu
rning-a-view-versus-a-copy
  schedule['wk1'] = list(zip(schedule.Wk1lat, schedule.Wk1long))
/var/folders/6p/v9rm9dqj4xq8113by2w3q15m0000gn/T/ipykernel_20020/3267419667.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#retu
rning-a-view-versus-a-copy
```

```
rning-a-view-versus-a-copy
  schedule['wk14'] = list(zip(schedule.Wk14lat, schedule.Wk14long))
/var/folders/6p/v9rm9dqj4xq8113by2w3q15m0000gn/T/ipykernel_20020/3267419667.py:17: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#retu
rning-a-view-versus-a-copy
  schedule['wk15'] = list(zip(schedule.Wk15lat, schedule.Wk15long))
```

GeoPy.distance function

Now, I was ready to start calculating distances. First, I created blank columns to store each distance traveled by a team, per week. Then, utilizing the geopy.distance function, I calculated the distance between every team's home stadium and the location of their game on a given week, multiplied by 2 (since teams would have to travel back home after the game), and stored these values into the corresponding blank column that I just generated. I used a for loop to iterate through the dataframe and calculate the distance formula for each team, week by week.

In [660…

```
#Create blank columns to store each distance traveled per week
schedule['wk0travel'] = ''
schedule['wk1travel'] = ''
schedule['wk2travel'] = ''
schedule['wk3travel'] = ''
schedule['wk4travel'] = ''
schedule['wk5travel'] = ''
schedule['wk6travel'] = ''
schedule['wk7travel'] = ''
schedule['wk8travel'] = ''
schedule['wk9travel'] = ''
schedule['wk10travel'] = ''
schedule['wk11travel'] = ''
schedule['wk12travel'] = ''
schedule['wk13travel'] = ''
schedule['wk14travel'] = ''
schedule['wk15travel'] = ''

#Find the distances traveled by each team every week
for i in range(0, len(schedule)):
    schedule.wk0travel.iloc[i] = round((dist(schedule.iloc[i]['coords'], schedule.iloc[i]['wk0']).mi),2)
    schedule.wk1travel.iloc[i] = round((dist(schedule.iloc[i]['coords'], schedule.iloc[i]['wk1']).mi),2)
    schedule.wk2travel.iloc[i] = round((dist(schedule.iloc[i]['coords'], schedule.iloc[i]['wk2']).mi),2)
    schedule.wk3travel.iloc[i] = round((dist(schedule.iloc[i]['coords'], schedule.iloc[i]['wk3']).mi),2)
    schedule.wk4travel.iloc[i] = round((dist(schedule.iloc[i]['coords'], schedule.iloc[i]['wk4']).mi),2)
    schedule.wk5travel.iloc[i] = round((dist(schedule.iloc[i]['coords'], schedule.iloc[i]['wk5']).mi),2)
    schedule.wk6travel.iloc[i] = round((dist(schedule.iloc[i]['coords'], schedule.iloc[i]['wk6']).mi),2)
    schedule.wk7travel.iloc[i] = round((dist(schedule.iloc[i]['coords'], schedule.iloc[i]['wk7']).mi),2)
    schedule.wk8travel.iloc[i] = round((dist(schedule.iloc[i]['coords'], schedule.iloc[i]['wk8']).mi),2)
    schedule.wk9travel.iloc[i] = round((dist(schedule.iloc[i]['coords'], schedule.iloc[i]['wk9']).mi),2)
    schedule.wk10travel.iloc[i] = round((dist(schedule.iloc[i]['coords'], schedule.iloc[i]['wk10']).mi),2)
    schedule.wk11travel.iloc[i] = round((dist(schedule.iloc[i]['coords'], schedule.iloc[i]['wk11']).mi),2)
    schedule.wk12travel.iloc[i] = round((dist(schedule.iloc[i]['coords'], schedule.iloc[i]['wk12']).mi),2)
    schedule.wk13travel.iloc[i] = round((dist(schedule.iloc[i]['coords'], schedule.iloc[i]['wk13']).mi),2)
    schedule.wk14travel.iloc[i] = round((dist(schedule.iloc[i]['coords'], schedule.iloc[i]['wk14']).mi),2)
    schedule.wk15travel.iloc[i] = round((dist(schedule.iloc[i]['coords'], schedule.iloc[i]['wk15']).mi),2)
```

```
/var/folders/6p/v9rm9dqj4xq8113by2w3q15m0000gn/T/ipykernel_20020/1576057175.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#retu
rning-a-view-versus-a-copy
  schedule['wk0travel'] = ''
/var/folders/6p/v9rm9dqj4xq8113by2w3q15m0000gn/T/ipykernel_20020/1576057175.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#retu
rning-a-view-versus-a-copy
  schedule['wk1travel'] = ''
/var/folders/6p/v9rm9dqj4xq8113by2w3q15m0000gn/T/ipykernel_20020/1576057175.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#retu
rning-a-view-versus-a-copy
  schedule['wk2travel'] = ''
/var/folders/6p/v9rm9dqj4xq8113by2w3q15m0000gn/T/ipykernel_20020/1576057175.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#retu
```

```
ServerApp.rate_limit_window=3.0 (secs)
```

travel dataframe

Once this loop was done running, I created another dataframe, travel, that contained the team, conference, and distance columns from the schedule dataframe. From there, I created a new column, 'travelSum', that summed up all distances from weeks 0-15 for each team, and multiplied this value by 2 to get the total travel distances for each team. This dataframe is shown below.

In [599…
```
#Create dataframe with team, conference, and sum of total travel *2 (to account for round trip distance))
travel = schedule.iloc[:, [0,1,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34]]
cols = ['wk0travel','wk1travel','wk2travel','wk3travel','wk4travel','wk5travel','wk6travel','wk7travel','wk8tra
travel['travelSum'] = round(travel[cols].sum(axis = 1)*2, 2)
travel = travel[['team','conference','travelSum']]
travel
```

```
/var/folders/6p/v9rm9dqj4xq8113by2w3q15m0000gn/T/ipykernel_20020/1568812640.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#retu
rning-a-view-versus-a-copy
  travel['travelSum'] = round(travel[cols].sum(axis = 1)*2, 2)
```

Out[599…

	team	conference	travelSum
0	Michigan	Big Ten	5204.54
1	Penn State	Big Ten	8621.4
2	Ohio State	Big Ten	5645.44
3	Texas A&M	SEC	5806.72
4	Tennessee	SEC	3649.86
...	...	...	...
129	Liberty	C-USA	8184.3
130	Sam Houston State	C-USA	7879.12
131	Coastal Carolina	Sun Belt	4576.7
132	Kennesaw State	C-USA	10230.5
133	UAB	American	5390.22

134 rows × 3 columns

Analysis

I decided to keep my analysis rather simple, and just focus on the top 20 and bottom 20 teams in terms of total travel, as well as average travel per conference. Top and bottom 20 teams are shown below, along with graphical depictions of each.

I expected Hawai'i to be the team that traveled the most, due to obvious reasons. However, I also expected there to be many more programs from the smaller conferences such as the MAC, C-USA, and Sun Belt on this list. These conferences typically load their out-of-conference schedules with "Buy Games", where they travel all over the country to play away games against bigger, better schools in exchange for massive paydays. The fact that there are only 3 schools from the C-USA, and none from the MAC or Sun Belt proves this theory wrong.

In [639…
```
#Find top 20 and bottom 20 teams in terms of miles traveled
traveltop20 = travel.sort_values(by = 'travelSum', ascending = False).head(20)
travelbtm20 = travel.sort_values(by = 'travelSum', ascending = True).head(20)
```

In [641…
```
traveltop20
```

Out[641…

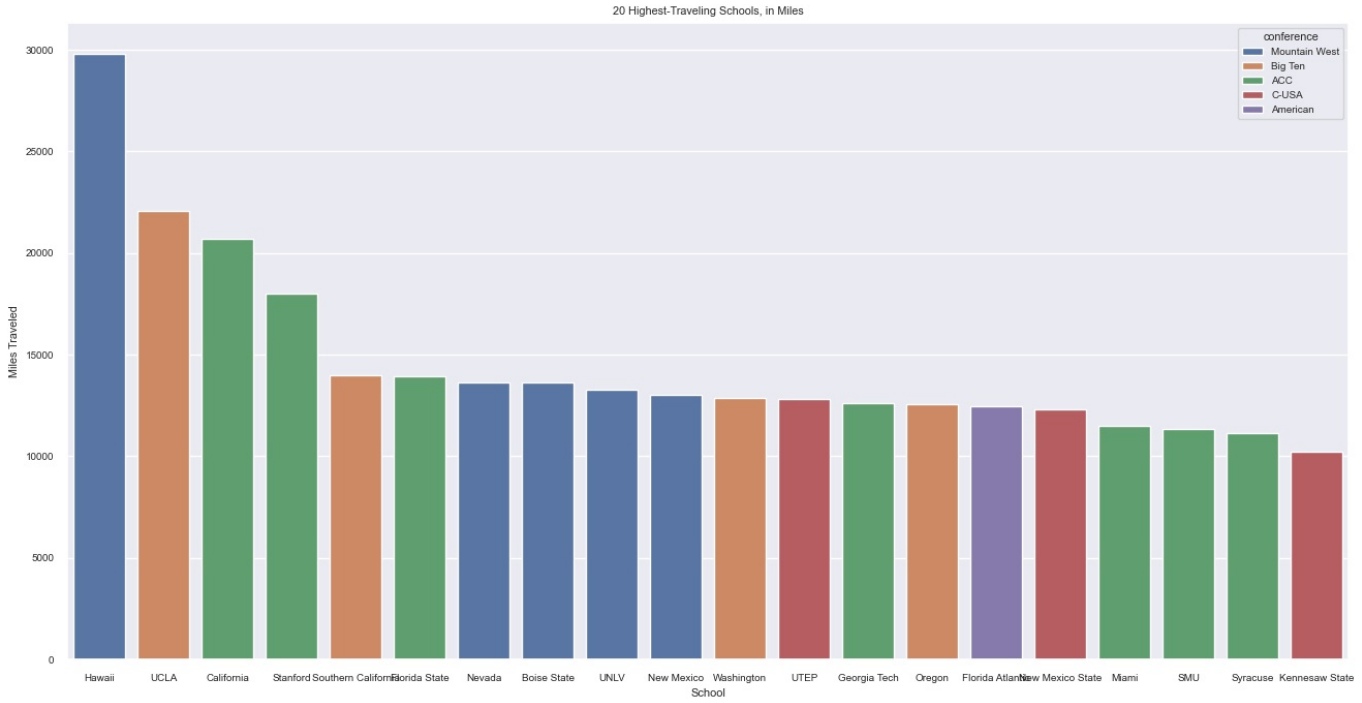
	team	conference	travelSum
59	Hawaii	Mountain West	29814.04
10	UCLA	Big Ten	22091.02
35	California	ACC	20699.64
62	Stanford	ACC	17995.04
8	Southern California	Big Ten	13993.32
14	Florida State	ACC	13950.4
103	Nevada	Mountain West	13646.04
81	Boise State	Mountain West	13613.5
83	UNLV	Mountain West	13275.0
78	New Mexico	Mountain West	13030.0
26	Washington	Big Ten	12853.18
56	UTEP	C-USA	12827.9
48	Georgia Tech	ACC	12639.6
50	Oregon	Big Ten	12548.3
106	Florida Atlantic	American	12442.16
100	New Mexico State	C-USA	12327.56
31	Miami	ACC	11511.68
92	SMU	ACC	11364.2
63	Syracuse	ACC	11161.2
132	Kennesaw State	C-USA	10230.5

In [643…

```
#Plot 20 highest travelers
plt.figure(figsize = (16,8))
sns.set(font_scale = 0.65)
plt.title('20 Highest-Traveling Schools, in Miles')
plt.xlabel('School', fontsize = 8)
plt.ylabel('Miles Traveled')
sns.barplot(traveltop20, x = 'team', y = 'travelSum', hue = 'conference')
```

Out[643…

```
<Axes: title={'center': '20 Highest-Traveling Schools, in Miles'}, xlabel='School', ylabel='Miles Traveled'>
```



In [645…

```
travelbtm20
```

Out[645…

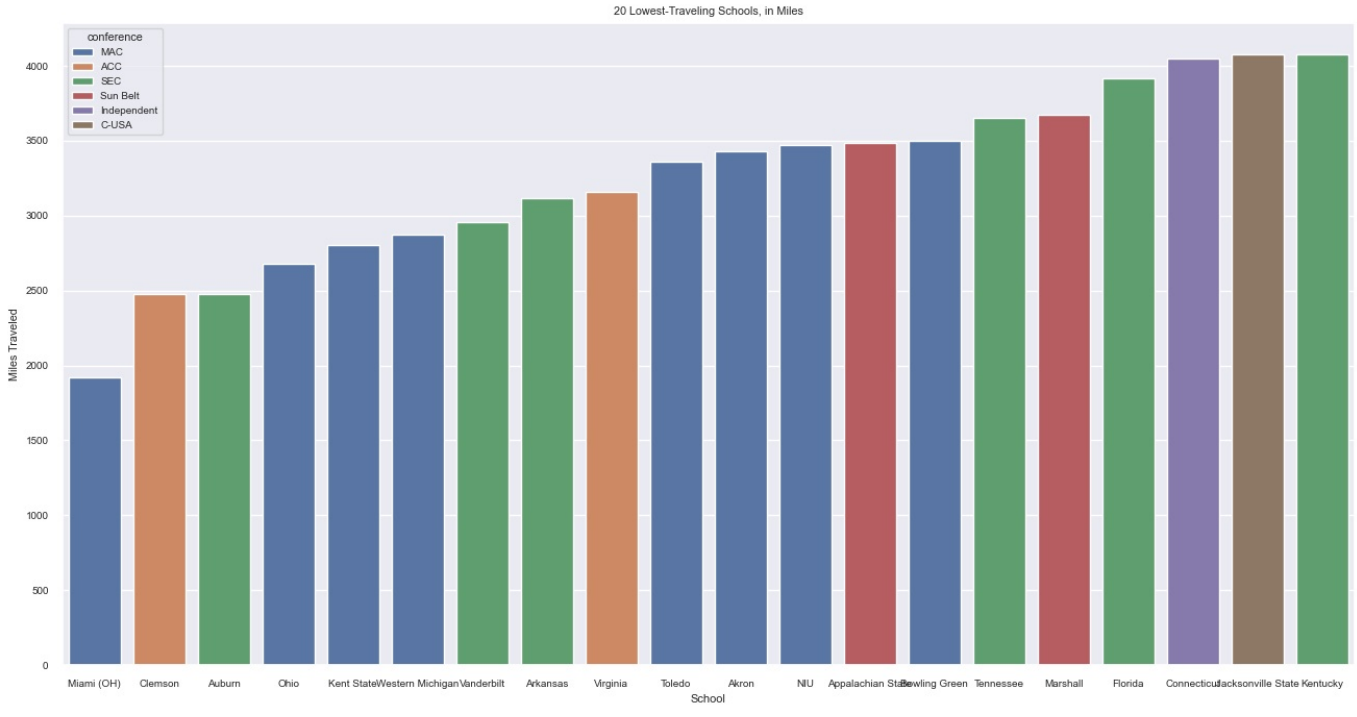
	team	conference	travelSum
117	Miami (OH)	MAC	1922.88
16	Clemson	ACC	2474.42
12	Auburn	SEC	2479.62
119	Ohio	MAC	2677.42
115	Kent State	MAC	2803.9
101	Western Michigan	MAC	2872.5
74	Vanderbilt	SEC	2958.4
21	Arkansas	SEC	3115.7
39	Virginia	ACC	3155.94
112	Toledo	MAC	3358.0
105	Akron	MAC	3431.22
114	NIU	MAC	3471.34
118	Appalachian State	Sun Belt	3483.56
120	Bowling Green	MAC	3496.3
4	Tennessee	SEC	3649.86
79	Marshall	Sun Belt	3675.76
11	Florida	SEC	3915.76
75	Connecticut	Independent	4050.98
128	Jacksonville State	C-USA	4074.92
37	Kentucky	SEC	4078.52

In [647…

```
#Plot 20 lowest travelers
plt.figure(figsize = (16,8))
plt.title('20 Lowest-Traveling Schools, in Miles')
sns.set(font_scale = 0.65)
plt.xlabel('School', fontsize = 8)
plt.ylabel('Miles Traveled')
sns.barplot(travelbtm20, x = 'team', y = 'travelSum', hue = 'conference')
```

Out[647…

```
<Axes: title={'center': '20 Lowest-Traveling Schools, in Miles'}, xlabel='School', ylabel='Miles Traveled'>
```

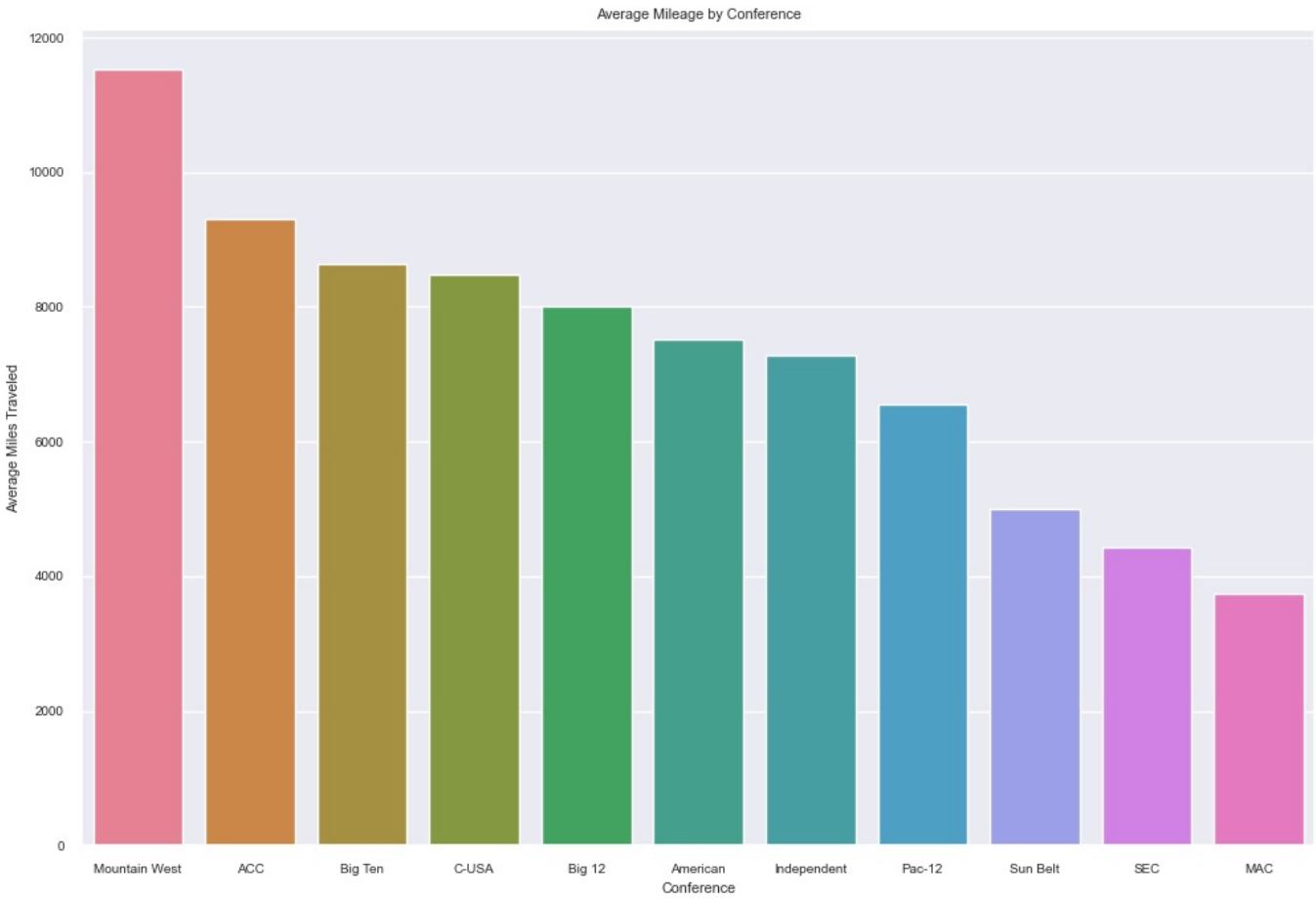


Average Mileage, per Conference

Here, I plotted the average mileage by conference. I hypothesized that the Mountain West would have the most travel due to Hawai'i being in this conference, but did not expect the ACC to be second in average travel. The offseason additions of California, Stanford, and SMU made an extreme impact on the travel schedules of the ACC, moreso than even the Big Ten, which added four west coast teams in Oregon, UCLA, USC, and Washington.

In [650…
```
#Plot average mileage by conference
conferences = pd.DataFrame(travel.groupby('conference')['travelSum'].mean().sort_values(ascending = False))
conferences
plt.figure(figsize = (12,8))
sns.barplot(conferences, x = 'conference', y = 'travelSum', hue = 'conference')
plt.title('Average Mileage by Conference')
plt.ylabel('Average Miles Traveled')
plt.xlabel('Conference')
```

Out[650…
```
Text(0.5, 0, 'Conference')
```



Next Steps

For the next part of this project, I am going to look at the schedules of each team from the 2023 season, before the massive conference realignment. I will then compare each team's total travel from last year to this year, and see where the biggest disparities lie. I predict that Hawaii and the Mountain West will once again dominate in total distance traveled, but expect to see a lot of the Top 20 travelers like UCLA, Stanford, Cal, and USC disappear from the Top 20 list from 2023, since their old conference opponents were much closer in proximity.

In [ ]:

In [ ]: