Econ 524 NCAA Import

This project is more difficult than the previous ones, in my opinion. Please don't wait too long to start it. You may work in groups of 2.

Overview

In this project, you will take raw data on college football game outcomes found on the internet and transform it into the format used in the past two projects for ranking teams. At the website:

https://wilson.engr.wisc.edu/rsfc/history/howell/cf2010gms.txt

you can see what the raw data looks like for the year 2010. More details on this below.

You should write a function (name it "ncaaimport") that takes a year (any between 1961 and 2013) as input and creates a data frame in the format that we used as input for the previous Colley assignments (i.e. the output data frame should have 4 columns: teams, wins, losses, and opponents). I will test your function by calling the provided colley_rank function (my solution for the Colley Matrix project) with the output of your ncaaimport function. Your function should get the data from web (described below) and should work without any data previously loaded into Python.

The data is available in a different file for every year at the website:

https://wilson.engr.wisc.edu/rsfc/history/howell/cf****gms.txt

where the **** should be replaced with the year. For example, for 2010:

https://wilson.engr.wisc.edu/rsfc/history/howell/cf2010gms.txt

The data files have the following 6 fields:

date, awayteam, awayscore, hometeam, homescore, location

Note: location is empty except for games that took place at a neutral site.

Tips

1. The data files on the web are "fixed width" files, meaning that each column is of a fixed number of characters. You can use the pd.read_fwf function in pandas to read this type of data. Provide the filename as the first argument (include the full string starting with "https://" for the filename). The second argument is a *list* that defines the field lengths. For example, first element of the list should be the tuple (0,10) which means the first field starts at position 0 and ends at position 9. You should investigate one file to see how many characters are used for each field. You should set header=None since the file does not have headers and use the names argument to provide a list of column names.

2. Recall that the + operator works on strings by concatenating them together. So for example, if the variable a=6, then the expression:

"The number after "+str(a-1)+" is "+str(a)+"."

Would evaluate to “The number after 5 is 6.” (Note: I am using the str function to convert integers to strings.) This will be useful for creating the filename to pass to read_fwf, specifically for filling in the **** in the filename above.

3. Some games ended in a tie. You should drop those games completely. Do not count them as a game played.

4. Some teams in a season played only a few games (like 1 or 2). That's because they are Division 2 teams that played most of their games against teams not being ranked. You should drop these teams and all of the games they played. Use the following heuristic. Check if a team played 6 or more games. Teams that played fewer than 6 games should be dropped. Furthermore, games against teams that played fewer than 6 games should also be dropped. For example, if Florida (a Division 1 team) played a game against a team that had no other games in the data set, you should drop the game from Florida's schedule - do not count it as a game played for Florida**. You should do this dropping procedure BEFORE dropping ties.**

What to Turn In

A file called ncaaimport.py that defines a function called ncaaimport. The function should take one argument (the year, of type int) and returns a data frame. The data frame should have 4 columns with the following names: team, wins, losses, and opponents (and should work if supplied as input to the colley_rank function I provided on Canvas in colley.py).

Algorithm

Here are the steps, in order, that I followed. All of these steps should be part of your function. Some of these can be re-arranged or combined, but I am providing as a guide in case it is helpful. Come see me if you need additional help.

1. download data for given year from the web into data frame using read_fwf

2. either manually or with help of pandas (i.e. groupby), collapse data to create a data frame with one observation per team that counts number of away games for each team

3. repeat step (2) for home games

4. merge data frames from steps (2) and (3) to compute total games played for each team

5. determine which teams to drop based on step (4) and drop all the relevant games from original data frame (i.e. drop all games from the step (1) data where one of the teams had fewer than 6 games played)

6. for remaining games after step (5), compute if it was a tie. drop if a tie

7. for remaining games after step (6), compute if home or away won.

8. re-do steps (2) and (3) but this time count wins in addition to number of games.

9. Also, keep track of all opponents for each team. Here's how I did this (this is probably the most difficult part of the project). Create a Series (or dictionary) mapping team names to id's (id's are made up by you but should be integers going from 0 to num_teams-1). Add columns for the home_id and away_id to the original data frame (the data frame with one row per game). Next, take a look at my example from the data wrangling script of using groupby.agg with the list function – this will help to create a list of opponents for each team.

10. combine into team name, wins, losses, opponents into data frame in correct format