cs109a_hw1

September 20, 2017

1 CS 109A/STAT 121A/AC 209A/CSCI E-109A: Homework 1

Harvard University Fall 2017 Instructors: Pavlos Protopapas, Kevin Rader, Rahul Dave, Margo Levine

---

1.0.1 INSTRUCTIONS

WARNING: There is web page scraping in this homework. It takes about 40 minutes. **Do not wait till the last minute** to do this homework.

- To submit your assignment follow the instructions given in canvas.
- Restart the kernel and run the whole notebook again before you submit. There is an important CAVEAT to this. DO NOT run the web-page fetching cells again. (We have provided hints like `# DO NOT RERUN THIS CELL WHEN SUBMITTING` on some of the cells where we provide the code). Instead load your data structures from the JSON files we will ask you to save below. Otherwise you will be waiting for a long time. (Another reason to not wait until the last moment to submit.)
- Do not include your name in the notebook.

---

2 Homework 1: Rihanna or Mariah?

Billboard Magazine puts out a top 100 list of "singles" every week. Information from this list, as well as that from music sales, radio, and other sources is used to determine a top-100 "singles" of the year list. A **single** is typically one song, but sometimes can be two songs which are on one "single" record.

In this homework you will:

1. Scrape Wikipedia to obtain infprmation about the best singers and groups from each year (distinguishing between the two groups) as determined by the Billboard top 100 charts. You will have to clean this data. Along the way you will learn how to save data in json files to avoid repeated scraping.

2. Scrape Wikipedia to obtain information on these singers. You will have to scrape the web pages, this time using a cache to guard against network timeouts (or your laptop going to sleep). You will again clean the data, and save it to a json file.
3. Use pandas to represent these two datasets and merge them.
4. Use the individual and merged datasets to visualize the performance of the artists and their songs. We have kept the amount of analysis limited here for reasons of time; but you might enjoy exploring music genres and other aspects of the music business you can find on these wikipedia pages at your own leisure.

You should have worked through Lab0 and Lab 1, and Lecture 2. Lab 2 will help as well.

As usual, first we import the necessary libraries. In particular, we use Seaborn to give us a nicer default color palette, with our plots being of large (`poster`) size and with a white-grid background.

```
In [1]: %matplotlib inline
        import functools
        import numpy as np
        import scipy as sp
        import matplotlib as mpl
        import matplotlib.cm as cm
        import matplotlib.pyplot as plt
        import pandas as pd
        import time
        pd.set_option('display.width', 500)
        pd.set_option('display.max_columns', 100)
        pd.set_option('display.notebook_repr_html', True)
        import seaborn as sns
```

2.1 Q1. Scraping Wikipedia for Billboard Top 100.

In this question you will scrape Wikipedia for the Billboard's top 100 singles.

2.1.1 Scraping Wikipedia for Billboard singles

We'll be using BeautifulSoup, and suggest that you use Python's built in `requests` library to fetch the web page.

1.1 Parsing the Billboard Wikipedia page for 1970 Obtain the web page at http://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_singles_of_1970 using a HTTP GET request. From this web page we'll extract the top 100 singles and their rankings. Create a list of dictionaries, 100 of them to be precise, with entries like

`{'url': '/wiki/Sugarloaf_(band)', 'ranking': 30, 'band_singer': 'Sugarloaf', 'title': 'Green-Eyed Lady'}`.

If you look at that web page, you'll see a link for every song, from which you can get the `url` of the singer or band. We will use these links later to scrape information about the singer or band. From the listing we can also get the band or singer name `band_singer`, and `title` of the song.

HINT: look for a table with class `wikitable`.

You should get something similar to this (where songs is the aforementioned list):

```
songs[2:4]
```

```
[{'band_singer': 'The Guess Who',
  'ranking': 3,
  'title': '"American Woman"',
  'url': '/wiki/The_Guess_Who'},
 {'band_singer': 'B.J. Thomas',
  'ranking': 4,
  'title': '"Raindrops Keep Fallin\' on My Head"',
  'url': '/wiki/B.J._Thomas'}]
```

In [2]:
```
import requests
from bs4 import BeautifulSoup
from IPython.display import IFrame, HTML
import time
```

In [3]:
```
# Get Wiki padge for 1970's top 100
req = requests.get(" http://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_singles_o
page = req.text
# Use Beautiful Soup to convert the HTML
soup = BeautifulSoup(page, 'html.parser')
# The classes of all tables that have a class sttribute set on them
[t["class"] for t in soup.find_all("table") if t.get("class")]
# Find the table with class-types 'sortable' and 'wikitable'
dfinder = lambda tag: tag.name=='table' and tag.get('class') == ['wikitable', 'sortable
table_songs = soup.find_all(dfinder)
```

In [4]:
```
# Extract rows from table_demographics
rows = [row for row in table_songs[0].find_all("tr")]

# Insert table data into dictionary
songs = []
for row in rows[1:]:
    entries = row.find_all("td")
    if entries[1].find("a"):
        songEntry = {'band_singer': entries[2].text, 'ranking': int(entries[0].text),
                'title': ('"' + entries[1].find("a").get("title") + '"'), 'url': entri
    # if song doesn't have URL
    else:
        songEntry = {'band_singer': entries[2].text, 'ranking': int(entries[0].text),
                'title': ("'" + entries[1].text + "'"), 'url': entries[2].find("a").ge
    songs.append(songEntry)
songs
```

Out[4]:
```
[{'band_singer': 'Simon & Garfunkel',
  'ranking': 1,
  'title': '"Bridge over Troubled Water (song)"',
  'url': '/wiki/Simon_%26_Garfunkel'},
 {'band_singer': 'The Carpenters',
```

```
 'ranking': 2,
 'title': '"(They Long to Be) Close to You"',
 'url': '/wiki/The_Carpenters'},
{'band_singer': 'The Guess Who',
 'ranking': 3,
 'title': '"American Woman"',
 'url': '/wiki/The_Guess_Who'},
{'band_singer': 'B.J. Thomas',
 'ranking': 4,
 'title': '"Raindrops Keep Fallin\' on My Head"',
 'url': '/wiki/B.J._Thomas'},
{'band_singer': 'Edwin Starr',
 'ranking': 5,
 'title': '"War (The Temptations song)"',
 'url': '/wiki/Edwin_Starr'},
{'band_singer': 'Diana Ross',
 'ranking': 6,
 'title': '"Ain\'t No Mountain High Enough"',
 'url': '/wiki/Diana_Ross'},
{'band_singer': 'The Jackson 5',
 'ranking': 7,
 'title': '"I\'ll Be There (The Jackson 5 song)"',
 'url': '/wiki/The_Jackson_5'},
{'band_singer': 'Rare Earth',
 'ranking': 8,
 'title': '"Get Ready (The Temptations song)"',
 'url': '/wiki/Rare_Earth_(band)'},
{'band_singer': 'The Beatles',
 'ranking': 9,
 'title': '"Let It Be (song)"',
 'url': '/wiki/The_Beatles'},
{'band_singer': 'Freda Payne',
 'ranking': 10,
 'title': '"Band of Gold (Freda Payne song)"',
 'url': '/wiki/Freda_Payne'},
{'band_singer': 'Three Dog Night',
 'ranking': 11,
 'title': '"Mama Told Me Not to Come"',
 'url': '/wiki/Three_Dog_Night'},
{'band_singer': 'Ray Stevens',
 'ranking': 12,
 'title': '"Everything Is Beautiful"',
 'url': '/wiki/Ray_Stevens'},
{'band_singer': 'Bread',
 'ranking': 13,
 'title': '"Make It with You"',
 'url': '/wiki/Bread_(band)'},
{'band_singer': 'Vanity Fare',
```

```
 'ranking': 14,
 'title': '"Hitchin\' a Ride (Vanity Fare song)"',
 'url': '/wiki/Vanity_Fare'},
{'band_singer': 'The Jackson 5',
 'ranking': 15,
 'title': '"ABC (The Jackson 5 song)"',
 'url': '/wiki/The_Jackson_5'},
{'band_singer': 'The Jackson 5',
 'ranking': 16,
 'title': '"The Love You Save"',
 'url': '/wiki/The_Jackson_5'},
{'band_singer': 'Neil Diamond',
 'ranking': 17,
 'title': '"Cracklin\' Rosie"',
 'url': '/wiki/Neil_Diamond'},
{'band_singer': 'Dawn',
 'ranking': 18,
 'title': '"Candida (song)"',
 'url': '/wiki/Tony_Orlando_and_Dawn'},
{'band_singer': 'Sly & the Family Stone',
 'ranking': 19,
 'title': '"Thank You (Falettinme Be Mice Elf Agin)"',
 'url': '/wiki/Sly_%26_the_Family_Stone'},
{'band_singer': 'Eric Burdon & War',
 'ranking': 20,
 'title': '"Spill the Wine"',
 'url': '/wiki/Eric_Burdon'},
{'band_singer': 'Five Stairsteps',
 'ranking': 21,
 'title': '"O-o-h Child"',
 'url': '/wiki/Five_Stairsteps'},
{'band_singer': 'Norman Greenbaum',
 'ranking': 22,
 'title': '"Spirit in the Sky"',
 'url': '/wiki/Norman_Greenbaum'},
{'band_singer': 'Melanie',
 'ranking': 23,
 'title': '"Lay Down (Candles in the Rain)"',
 'url': '/wiki/Melanie_Safka'},
{'band_singer': 'The Temptations',
 'ranking': 24,
 'title': '"Ball of Confusion (That\'s What the World Is Today)"',
 'url': '/wiki/The_Temptations'},
{'band_singer': 'The Moments',
 'ranking': 25,
 'title': '"Love on a Two-Way Street"',
 'url': '/wiki/Ray,_Goodman_%26_Brown'},
{'band_singer': 'The Poppy Family',
```

```
 'ranking': 26,
 'title': '"Which Way You Goin\' Billy? (song)"',
 'url': '/wiki/The_Poppy_Family'},
{'band_singer': 'Free',
 'ranking': 27,
 'title': '"All Right Now"',
 'url': '/wiki/Free_(band)'},
{'band_singer': 'The Jackson 5',
 'ranking': 28,
 'title': '"I Want You Back"',
 'url': '/wiki/The_Jackson_5'},
{'band_singer': 'Bobby Sherman',
 'ranking': 29,
 'title': '"Julie, Do Ya Love Me"',
 'url': '/wiki/Bobby_Sherman'},
{'band_singer': 'Sugarloaf',
 'ranking': 30,
 'title': '"Green-Eyed Lady"',
 'url': '/wiki/Sugarloaf_(band)'},
{'band_singer': 'Stevie Wonder',
 'ranking': 31,
 'title': '"Signed, Sealed, Delivered I\'m Yours"',
 'url': '/wiki/Stevie_Wonder'},
{'band_singer': 'Blues Image',
 'ranking': 32,
 'title': '"Ride Captain Ride"',
 'url': '/wiki/Blues_Image'},
{'band_singer': 'Shocking Blue',
 'ranking': 33,
 'title': '"Venus (Shocking Blue song)"',
 'url': '/wiki/Shocking_Blue'},
{'band_singer': 'John Lennon',
 'ranking': 34,
 'title': '"Instant Karma!"',
 'url': '/wiki/John_Lennon'},
{'band_singer': 'Clarence Carter',
 'ranking': 35,
 'title': '"Patches (Chairmen of the Board song)"',
 'url': '/wiki/Clarence_Carter'},
{'band_singer': 'Creedence Clearwater Revival',
 'ranking': 36,
 'title': '"Lookin\' out My Back Door"',
 'url': '/wiki/Creedence_Clearwater_Revival'},
{'band_singer': 'Brook Benton',
 'ranking': 37,
 'title': '"Rainy Night in Georgia"',
 'url': '/wiki/Brook_Benton'},
{'band_singer': 'Kenny Rogers & The First Edition',
```

```
 'ranking': 38,
 'title': '\'"Something\'s Burning"\'',
 'url': '/wiki/Kenny_Rogers'},
{'band_singer': 'Chairmen of the Board',
 'ranking': 39,
 'title': '"Give Me Just a Little More Time"',
 'url': '/wiki/Chairmen_of_the_Board'},
{'band_singer': 'Edison Lighthouse',
 'ranking': 40,
 'title': '"Love Grows (Where My Rosemary Goes)"',
 'url': '/wiki/Edison_Lighthouse'},
{'band_singer': 'The Beatles',
 'ranking': 41,
 'title': '"The Long and Winding Road"',
 'url': '/wiki/The_Beatles'},
{'band_singer': 'Anne Murray',
 'ranking': 42,
 'title': '"Snowbird (song)"',
 'url': '/wiki/Anne_Murray'},
{'band_singer': 'Marmalade',
 'ranking': 43,
 'title': '"Reflections of My Life"',
 'url': '/wiki/Marmalade_(band)'},
{'band_singer': 'Eddie Holman',
 'ranking': 44,
 'title': '"Hey There Lonely Girl"',
 'url': '/wiki/Eddie_Holman'},
{'band_singer': 'The Jaggerz',
 'ranking': 45,
 'title': '"The Rapper"',
 'url': '/wiki/The_Jaggerz'},
{'band_singer': 'The Hollies',
 'ranking': 46,
 'title': '"He Ain\'t Heavy, He\'s My Brother"',
 'url': '/wiki/The_Hollies'},
{'band_singer': "Alive N Kickin'",
 'ranking': 47,
 'title': '"Tighter, Tighter"',
 'url': '/wiki/Alive_N_Kickin%27'},
{'band_singer': 'Badfinger',
 'ranking': 48,
 'title': '"Come and Get It (Badfinger song)"',
 'url': '/wiki/Badfinger'},
{'band_singer': 'Simon & Garfunkel',
 'ranking': 49,
 'title': '"Cecilia (Simon & Garfunkel song)"',
 'url': '/wiki/Simon_%26_Garfunkel'},
{'band_singer': 'Charles Wright & the Watts 103rd Street Rhythm Band',
```

```
 'ranking': 50,
 'title': '"Love Land (song)"',
 'url': '/wiki/Charles_Wright_%26_the_Watts_103rd_Street_Rhythm_Band'},
{'band_singer': 'Tyrone Davis',
 'ranking': 51,
 'title': '"Turn Back the Hands of Time (song)"',
 'url': '/wiki/Tyrone_Davis'},
{'band_singer': 'The Kinks',
 'ranking': 52,
 'title': '"Lola (song)"',
 'url': '/wiki/The_Kinks'},
{'band_singer': 'Mungo Jerry',
 'ranking': 53,
 'title': '"In the Summertime (Mungo Jerry song)"',
 'url': '/wiki/Mungo_Jerry'},
{'band_singer': 'R. Dean Taylor',
 'ranking': 54,
 'title': '"Indiana Wants Me"',
 'url': '/wiki/R._Dean_Taylor'},
{'band_singer': 'Rare Earth',
 'ranking': 55,
 'title': '"(I Know) I\'m Losing You"',
 'url': '/wiki/Rare_Earth_(band)'},
{'band_singer': 'Bobby Sherman',
 'ranking': 56,
 'title': '"Easy Come, Easy Go (Bobby Sherman song)"',
 'url': '/wiki/Bobby_Sherman'},
{'band_singer': 'Charles Wright & the Watts 103rd Street Rhythm Band',
 'ranking': 57,
 'title': '"Express Yourself (Charles Wright & the Watts 103rd Street Rhythm Band son
 'url': '/wiki/Charles_Wright_%26_the_Watts_103rd_Street_Rhythm_Band'},
{'band_singer': 'The Four Tops',
 'ranking': 58,
 'title': '"Still Water (Love)"',
 'url': '/wiki/The_Four_Tops'},
{'band_singer': 'Chicago',
 'ranking': 59,
 'title': '"Make Me Smile"',
 'url': '/wiki/Chicago_(band)'},
{'band_singer': 'Frijid Pink',
 'ranking': 60,
 'title': '"The House of the Rising Sun"',
 'url': '/wiki/Frijid_Pink'},
{'band_singer': 'Chicago',
 'ranking': 61,
 'title': '"25 or 6 to 4"',
 'url': '/wiki/Chicago_(band)'},
{'band_singer': 'White Plains',
```

```
 'ranking': 62,
 'title': '"My Baby Loves Lovin\'"',
 'url': '/wiki/White_Plains_(band)'},
{'band_singer': 'The Friends of Distinction',
 'ranking': 63,
 'title': '"Love or Let Me Be Lonely"',
 'url': '/wiki/The_Friends_of_Distinction'},
{'band_singer': 'The Brotherhood of Man',
 'ranking': 64,
 'title': '"United We Stand (song)"',
 'url': '/wiki/The_Brotherhood_of_Man'},
{'band_singer': 'The Carpenters',
 'ranking': 65,
 'title': '"We\'ve Only Just Begun"',
 'url': '/wiki/The_Carpenters'},
{'band_singer': 'Mark Lindsay',
 'ranking': 66,
 'title': '"Arizona (song)"',
 'url': '/wiki/Mark_Lindsay'},
{'band_singer': 'James Taylor',
 'ranking': 67,
 'title': '"Fire and Rain (song)"',
 'url': '/wiki/James_Taylor'},
{'band_singer': 'Gene Chandler',
 'ranking': 68,
 'title': '"Groovy Situation"',
 'url': '/wiki/Gene_Chandler'},
{'band_singer': 'Santana',
 'ranking': 69,
 'title': '"Evil Ways"',
 'url': '/wiki/Santana_(band)'},
{'band_singer': 'The Guess Who',
 'ranking': 70,
 'title': '"No Time (The Guess Who song)"',
 'url': '/wiki/The_Guess_Who'},
{'band_singer': 'The Delfonics',
 'ranking': 71,
 'title': '"Didn\'t I (Blow Your Mind This Time)"',
 'url': '/wiki/The_Delfonics'},
{'band_singer': 'Elvis Presley',
 'ranking': 72,
 'title': '"The Wonder of You"',
 'url': '/wiki/Elvis_Presley'},
{'band_singer': 'Creedence Clearwater Revival',
 'ranking': 73,
 'title': '"Up Around the Bend"',
 'url': '/wiki/Creedence_Clearwater_Revival'},
{'band_singer': 'Ronnie Dyson',
```

```
 'ranking': 74,
 'title': '"(If You Let Me Make Love To You Then) Why Can\'t I Touch You?"',
 'url': '/wiki/Ronnie_Dyson'},
{'band_singer': 'B.J. Thomas',
 'ranking': 75,
 'title': '"I Just Can\'t Help Believing"',
 'url': '/wiki/B.J._Thomas'},
{'band_singer': 'The Spinners',
 'ranking': 76,
 'title': '"It\'s a Shame (The Spinners song)"',
 'url': '/wiki/The_Spinners_(American_band)'},
{'band_singer': 'Bobbi Martin',
 'ranking': 77,
 'title': '"For the Love of Him"',
 'url': '/wiki/Bobbi_Martin'},
{'band_singer': 'Mountain',
 'ranking': 78,
 'title': '"Mississippi Queen"',
 'url': '/wiki/Mountain_(band)'},
{'band_singer': 'Ike & Tina Turner',
 'ranking': 79,
 'title': '"I Want to Take You Higher"',
 'url': '/wiki/Ike_%26_Tina_Turner'},
{'band_singer': 'Joe Cocker',
 'ranking': 80,
 'title': '"The Letter (The Box Tops song)"',
 'url': '/wiki/Joe_Cocker'},
{'band_singer': 'Tee Set',
 'ranking': 81,
 'title': '"Ma Belle Amie"',
 'url': '/wiki/Tee_Set'},
{'band_singer': 'The Originals',
 'ranking': 82,
 'title': '"The Bells (The Originals song)"',
 'url': '/wiki/The_Originals_(band)'},
{'band_singer': 'Christie',
 'ranking': 83,
 'title': '"Yellow River (song)"',
 'url': '/wiki/Christie_(band)'},
{'band_singer': '100 Proof (Aged in Soul)',
 'ranking': 84,
 'title': '"Somebody\'s Been Sleeping"',
 'url': '/wiki/100_Proof_(Aged_in_Soul)'},
{'band_singer': 'The Ides of March',
 'ranking': 85,
 'title': '"Vehicle (song)"',
 'url': '/wiki/The_Ides_of_March_(band)'},
{'band_singer': 'The Pipkins',
```

```
 'ranking': 86,
 'title': '"Gimme Dat Ding (song)"',
 'url': '/wiki/The_Pipkins'},
{'band_singer': 'Robin McNamara',
 'ranking': 87,
 'title': '"Lay a Little Lovin\' on Me"',
 'url': '/wiki/Robin_McNamara'},
{'band_singer': 'The Supremes',
 'ranking': 88,
 'title': '"Up the Ladder to the Roof"',
 'url': '/wiki/The_Supremes'},
{'band_singer': 'Creedence Clearwater Revival',
 'ranking': 89,
 'title': '"Travelin\' Band"',
 'url': '/wiki/Creedence_Clearwater_Revival'},
{'band_singer': 'The Sandpipers',
 'ranking': 90,
 'title': '"Come Saturday Morning (song)"',
 'url': '/wiki/The_Sandpipers'},
{'band_singer': 'The Temptations',
 'ranking': 91,
 'title': '"Psychedelic Shack (song)"',
 'url': '/wiki/The_Temptations'},
{'band_singer': 'Tom Jones',
 'ranking': 92,
 'title': '"Without Love (There Is Nothing)"',
 'url': '/wiki/Tom_Jones_(singer)'},
{'band_singer': 'Pacific Gas & Electric',
 'ranking': 93,
 'title': '"Are You Ready? (Pacific Gas & Electric song)"',
 'url': '/wiki/Pacific_Gas_%26_Electric_(band)'},
{'band_singer': 'Crosby, Stills, Nash & Young',
 'ranking': 94,
 'title': '"Woodstock (song)"',
 'url': '/wiki/Crosby,_Stills,_Nash_%26_Young'},
{'band_singer': 'Dionne Warwick',
 'ranking': 95,
 'title': '"I\'ll Never Fall in Love Again"',
 'url': '/wiki/Dionne_Warwick'},
{'band_singer': 'The New Seekers',
 'ranking': 96,
 'title': '\'"Look What They\'ve Done to My Song Ma"\'',
 'url': '/wiki/The_New_Seekers'},
{'band_singer': 'Joe South',
 'ranking': 97,
 'title': '"Walk A Mile In My Shoes"',
 'url': '/wiki/Joe_South'},
{'band_singer': 'B.B. King',
```

```
 'ranking': 98,
 'title': '"The Thrill Is Gone (1951 song)"',
 'url': '/wiki/B.B._King'},
{'band_singer': 'Glen Campbell',
 'ranking': 99,
 'title': '"It\'s Only Make Believe"',
 'url': '/wiki/Glen_Campbell'},
{'band_singer': 'Aretha Franklin',
 'ranking': 100,
 'title': '"Call Me (Aretha Franklin song)"',
 'url': '/wiki/Aretha_Franklin'}]
```

1.2 Generalize the previous: scrape Wikipedia from 1992 to 2014 By visiting the urls similar to the ones for 1970, we can obtain the billboard top 100 for the years 1992 to 2014. (We choose these later years rather than 1970 as you might find music from this era more interesting.) Download these using Python's `requests` module and store the text from those requests in a dictionary called `yearstext`. This dictionary ought to have as its keys the years (as integers from 1992 to 2014), and as values corresponding to these keys the text of the page being fetched.

You ought to sleep a second (look up `time.sleep` in Python) at the very least in-between fetching each web page: you do not want Wikipedia to think you are a marauding bot attempting to mount a denial-of-service attack.

HINT: you might find `range` and string-interpolation useful to construct the URLs .

```
In [5]: years = range(1992, 2015)
        text = []
        for year in range(1992, 2015):
            req = requests.get(" http://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_single
            text.append(req.text)
            time.sleep(1)
        yearstext = dict(zip(years,text))
```

1.3 Parse and Clean data Remember the code you wrote to get data from 1970 which produces a list of dictionaries, one corresponding to each single. Now write a function `parse_year(the_year, yeartext_dict)` which takes the year, prints it out, gets the text for the year from the just created `yearstext` dictionary, and return a list of dictionaries for that year, with one dictionary for each single. Store this list in the variable `yearinfo`.

The dictionaries **must** be of this form:

```
{'band_singer': ['Brandy', 'Monica'],
  'ranking': 2,
  'song': ['The Boy Is Mine'],
  'songurl': ['/wiki/The_Boy_Is_Mine_(song)'],
  'titletext': '" The Boy Is Mine "',
  'url': ['/wiki/Brandy_Norwood', '/wiki/Monica_(entertainer)']}
```

The spec of this function is provided below:

In [6]:
```
"""
Function
--------
parse_year

Inputs
------
the_year: the year you want the singles for
yeartext_dict: a dictionary with keys as integer years and values the downloaded web p
    from wikipedia for that year.

Returns
-------

a list of dictionaries, each of which corresponds to a single and has the
following data:

Eg:

{'band_singer': ['Brandy', 'Monica'],
  'ranking': 2,
  'song': ['The Boy Is Mine'],
  'songurl': ['/wiki/The_Boy_Is_Mine_(song)'],
  'titletext': '" The Boy Is Mine "',
  'url': ['/wiki/Brandy_Norwood', '/wiki/Monica_(entertainer)']}

A dictionary with the following data:
    band_singer: a list of bands/singers who made this single
    song: a list of the titles of songs on this single
    songurl: a list of the same size as song which has urls for the songs on the singl
        (see point 3 above)
    ranking: ranking of the single
    titletext: the contents of the table cell
    band_singer: a list of bands or singers on this single
    url: a list of wikipedia singer/band urls on this single: only put in the part
        of the url from /wiki onwards


Notes
-----
See description and example above.
"""
def parse_year(the_year, yeartext_dict):
    page =  yeartext_dict[the_year]
    soup = BeautifulSoup(page, 'html.parser')
    # Find the table with class-types 'sortable' and 'wikitable'
    table_songs = soup.find_all(dfinder)
    # Extract rows from table_demographics
```

```
        rows = [row for row in table_songs[0].find_all("tr")]
        # Insert table data into dictionary
        songs = []
        counter = 0
        for row in rows[1:]:
            counter += 1
            entries = row.find_all("td")
            # if song has URL
            if entries[0].find("a"):
                songurl = list(map(lambda x: x.get("href"), entries[0].find_all("a")))
                song = list(map(lambda x: x.text, entries[0].find_all("a")))
                titletext = functools.reduce((lambda x, y: x +" / "+ y), (list(map(lambda x
            else:
                songurl = [None]
                song = [entries[0].text.replace('"', '')]
                titletext = [entries[0].text]
            # if artist has URL
            if entries[1].find("a"):
                url = list(map(lambda x: x.get("href"), entries[1].find_all("a")))
                band_singer = list(map(lambda x: x.text, entries[1].find_all("a")))
            else:
                url = [None]
                band_singer = [entries[1].text]
            songEntry = {'band_singer': band_singer,'ranking': counter,'song': song,'songu
            songs.append(songEntry)
        return songs

    yearinfo = []

    for year in years:
        info = parse_year(year, yearstext)
        yearinfo.append(info)
```

Helpful notes Notice that some singles might have multiple songs:

```
{'band_singer': ['Jewel'],
  'ranking': 2,
  'song': ['Foolish Games', 'You Were Meant for Me'],
  'songurl': ['/wiki/Foolish_Games',
   '/wiki/You_Were_Meant_for_Me_(Jewel_song)'],
  'titletext': '" Foolish Games " / " You Were Meant for Me "',
  'url': ['/wiki/Jewel_(singer)']}
```

And some singles don't have a song URL:

```
{'band_singer': [u'Nu Flavor'],
  'ranking': 91,
```

```
 'song': [u'Heaven'],
 'songurl': [None],
 'titletext': u'"Heaven"',
 'url': [u'/wiki/Nu_Flavor']}
```

Thus there are some issues this function must handle:

1. There can be more than one `band_singer` as can be seen above (sometimes with a comma, sometimes with "featuring" in between). The best way to parse these is to look for the urls.
2. There can be two songs in a single, because of the way the industry works: there are two-sided singles. See https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_singles_of_1997 for an example. You can find other examples in 1998 and 1999.
3. The `titletext` is the contents of the table cell, and retains the quotes that Wikipedia puts on the single.
4. If no song anchor is found (see the 24th song in the above url), assume there is one song in the single, set `songurl` to [None] and the song name to the contents of the table cell with the quotes stripped (ie `song` is a one-element list with this the `titletext` stripped of its quotes).

As a check, we can do this for 1997. We'll print the first 5 outputs: `parse_year(1997, yearstext)[:5]`

This should give the following. Notice that the year 1997 exercises the edge cases we talked about earlier.

```
[{'band_singer': ['Elton John'],
  'ranking': 1,
  'song': ['Something About the Way You Look Tonight',
   'Candle in the Wind 1997'],
  'songurl': ['/wiki/Something_About_the_Way_You_Look_Tonight',
   '/wiki/Candle_in_the_Wind_1997'],
  'titletext': '" Something About the Way You Look Tonight " / " Candle in the Wind 1997 "',
  'url': ['/wiki/Elton_John']},
 {'band_singer': ['Jewel'],
  'ranking': 2,
  'song': ['Foolish Games', 'You Were Meant for Me'],
  'songurl': ['/wiki/Foolish_Games',
   '/wiki/You_Were_Meant_for_Me_(Jewel_song)'],
  'titletext': '" Foolish Games " / " You Were Meant for Me "',
  'url': ['/wiki/Jewel_(singer)']},
 {'band_singer': ['Puff Daddy', 'Faith Evans', '112'],
  'ranking': 3,
  'song': ["I'll Be Missing You"],
  'songurl': ['/wiki/I%27ll_Be_Missing_You'],
  'titletext': '" I\'ll Be Missing You "',
  'url': ['/wiki/Sean_Combs', '/wiki/Faith_Evans', '/wiki/112_(band)']},
 {'band_singer': ['Toni Braxton'],
  'ranking': 4,
  'song': ['Un-Break My Heart'],
  'songurl': ['/wiki/Un-Break_My_Heart'],
```

```
  'titletext': '" Un-Break My Heart "',
  'url': ['/wiki/Toni_Braxton']},
 {'band_singer': ['Puff Daddy', 'Mase'],
  'ranking': 5,
  'song': ["Can't Nobody Hold Me Down"],
  'songurl': ['/wiki/Can%27t_Nobody_Hold_Me_Down'],
  'titletext': '" Can\'t Nobody Hold Me Down "',
  'url': ['/wiki/Sean_Combs', '/wiki/Mase']}]
```

```
In [7]: parse_year(1997, yearstext)[:5]
```

```
Out[7]: [{'band_singer': ['Elton John'],
          'ranking': 1,
          'song': ['Something About the Way You Look Tonight',
           'Candle in the Wind 1997'],
          'songurl': ['/wiki/Something_About_the_Way_You_Look_Tonight',
           '/wiki/Candle_in_the_Wind_1997'],
          'titletext': '"Something About the Way You Look Tonight" / "Candle in the Wind 1997"
          'url': ['/wiki/Elton_John']},
         {'band_singer': ['Jewel'],
          'ranking': 2,
          'song': ['Foolish Games', 'You Were Meant for Me'],
          'songurl': ['/wiki/Foolish_Games',
           '/wiki/You_Were_Meant_for_Me_(Jewel_song)'],
          'titletext': '"Foolish Games" / "You Were Meant for Me (Jewel song)"',
          'url': ['/wiki/Jewel_(singer)']},
         {'band_singer': ['Puff Daddy', 'Faith Evans', '112'],
          'ranking': 3,
          'song': ["I'll Be Missing You"],
          'songurl': ['/wiki/I%27ll_Be_Missing_You'],
          'titletext': '"I\'ll Be Missing You"',
          'url': ['/wiki/Sean_Combs', '/wiki/Faith_Evans', '/wiki/112_(band)']},
         {'band_singer': ['Toni Braxton'],
          'ranking': 4,
          'song': ['Un-Break My Heart'],
          'songurl': ['/wiki/Un-Break_My_Heart'],
          'titletext': '"Un-Break My Heart"',
          'url': ['/wiki/Toni_Braxton']},
         {'band_singer': ['Puff Daddy', 'Mase'],
          'ranking': 5,
          'song': ["Can't Nobody Hold Me Down"],
          'songurl': ['/wiki/Can%27t_Nobody_Hold_Me_Down'],
          'titletext': '"Can\'t Nobody Hold Me Down"',
          'url': ['/wiki/Sean_Combs', '/wiki/Mase']}]
```

Save a json file of information from the scraped files We do not want to lose all this work, so let's save the last data structure we created to disk. That way if you need to re-run from here, you don't need to redo all these requests and parsing.