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

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

---

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.

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 (http://stanford.edu/~mwaskom/software/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 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
sns.set_style("whitegrid")
sns.set_context("poster")
```

Q1. Scraping Wikipedia for Billboard Top 100.

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

Scraping Wikipedia for Billboard singles

We'll be using BeautifulSoup (http://www.crummy.com/software/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 (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`.

In [2]:

```
import requests
from IPython.display import HTML
from bs4 import BeautifulSoup
req = requests.get("https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_sin
gles_of_1970")
page = req.text
soup = BeautifulSoup(page, 'html.parser')
table_100songs = soup.find("table", "wikitable")
songs = []
headrow = table_100songs.find("tr")
rows = table_100songs.find_all("tr")[1:]

for row in rows:
    col = row.find_all("td")
    cur_dict = dict()
    cur_dict["band_singer"]  = col[2].text
    cur_dict["ranking"] = col[0].text
    cur_dict["title"] = col[1].text
    cur_dict["url"] = col[2].find("a")["href"]
    songs.append(cur_dict)

songs
```

Out[2]:


```
[{'band_singer': 'Simon & Garfunkel',
  'ranking': '1',
  'title': '"Bridge over Troubled Water"',
  '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"',
  '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"',
 'url': '/wiki/The_Jackson_5'},
{'band_singer': 'Rare Earth',
 'ranking': '8',
 'title': '"Get Ready"',
 'url': '/wiki/Rare_Earth_(band)'},
{'band_singer': 'The Beatles',
 'ranking': '9',
 'title': '"Let It Be"',
 'url': '/wiki/The_Beatles'},
{'band_singer': 'Freda Payne',
 'ranking': '10',
 'title': '"Band of Gold"',
 '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"',
 'url': '/wiki/Vanity_Fare'},
{'band_singer': 'The Jackson 5',
 'ranking': '15',
 'title': '"ABC"',
 '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"',
 '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?"',
 '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"',
 '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"',
 '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"',
 '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"',
  'url': '/wiki/Badfinger'},
 {'band_singer': 'Simon & Garfunkel',
  'ranking': '49',
  'title': '"Cecilia"',
  'url': '/wiki/Simon_%26_Garfunkel'},
 {'band_singer': 'Charles Wright & the Watts 103rd Street Rhythm Ban
d',
  'ranking': '50',
  'title': '"Love Land"',
  'url': '/wiki/Charles_Wright_%26_the_Watts_103rd_Street_Rhythm_Ban
d'},
 {'band_singer': 'Tyrone Davis',
  'ranking': '51',
  'title': '"Turn Back the Hands of Time"',
  'url': '/wiki/Tyrone_Davis'},
 {'band_singer': 'The Kinks',
  'ranking': '52',
  'title': '"Lola"',
  'url': '/wiki/The_Kinks'},
 {'band_singer': 'Mungo Jerry',
  'ranking': '53',
  'title': '"In the Summertime"',
  '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"',
  'url': '/wiki/Bobby_Sherman'},
 {'band_singer': 'Charles Wright & the Watts 103rd Street Rhythm Ban
d',
  'ranking': '57',
  'title': '"Express Yourself"',
  'url': '/wiki/Charles_Wright_%26_the_Watts_103rd_Street_Rhythm_Ban
d'},
 {'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"',
 '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"',
 'url': '/wiki/Mark_Lindsay'},
{'band_singer': 'James Taylor',
 'ranking': '67',
 'title': '"Fire and Rain"',
 '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"',
 '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 Touc
h 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"',
  '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"',
  '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"',
  'url': '/wiki/The_Originals_(band)'},
{'band_singer': 'Christie',
  'ranking': '83',
  'title': '"Yellow River"',
  '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"',
```

```
 'url': '/wiki/The_Ides_of_March_(band)'},
{'band_singer': 'The Pipkins',
 'ranking': '86',
 'title': '"Gimme Dat Ding"',
 '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"',
 'url': '/wiki/The_Sandpipers'},
{'band_singer': 'The Temptations',
 'ranking': '91',
 'title': '"Psychedelic Shack"',
 '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?"',
 'url': '/wiki/Pacific_Gas_%26_Electric_(band)'},
{'band_singer': 'Crosby, Stills, Nash & Young',
 'ranking': '94',
 'title': '"Woodstock"',
 '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"',
 '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"',
 'url': '/wiki/Aretha_Franklin'}]
```

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'}]
```

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 [3]:

```
yearstext = dict()
for year in range(1992, 2015):
    request_str = "https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_sing
les_of_" + str(year)
    print("getting " + request_str)
    yearstext[year] = requests.get(request_str).text
    time.sleep(1)
```

```
getting https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_sin
gles_of_1992
getting https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_sin
gles_of_1993
getting https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_sin
gles_of_1994
getting https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_sin
gles_of_1995
getting https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_sin
gles_of_1996
getting https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_sin
gles_of_1997
getting https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_sin
gles_of_1998
getting https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_sin
gles_of_1999
getting https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_sin
gles_of_2000
getting https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_sin
gles_of_2001
getting https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_sin
gles_of_2002
getting https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_sin
gles_of_2003
getting https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_sin
gles_of_2004
getting https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_sin
gles_of_2005
getting https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_sin
gles_of_2006
getting https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_sin
gles_of_2007
getting https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_sin
gles_of_2008
getting https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_sin
gles_of_2009
getting https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_sin
gles_of_2010
getting https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_sin
gles_of_2011
getting https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_sin
gles_of_2012
getting https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_sin
gles_of_2013
getting https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_sin
gles_of_2014
```

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 [4]:

```
def parse_year(the_year, yeartext_dict):
    """
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 pages
    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
```

```
single

        (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 pa
rt
        of the url from /wiki onwards


Notes
-----
See description and example above.
    """
    print(the_year)
    dicts = []
    soup = BeautifulSoup(yeartext_dict[the_year], 'html.parser')
    table_100songs = soup.find("table", "wikitable")
    rows = table_100songs.find_all("tr")[1:]
    for row in rows:
        year_dict = dict()
        ranking = row.find("th").text
        col = row.find_all("td")
        assert(len(col) == 2)
        singers = col[1].find_all("a")
        songs = col[0].find_all("a")
        year_dict["band_singer"] = [s.text for s in singers]
        year_dict["ranking"] = int(ranking)
        if len(songs) == 0:
            year_dict["song"] = col[0].text
            year_dict["songurl"] = None
        else:
            year_dict["song"] = [s.text for s in songs]
            year_dict["songurl"] = [s["href"] for s in songs]
        year_dict["titletext"] = col[0].text

        year_dict["url"] = [s["href"] for s in singers]
        dicts.append(year_dict)

    return dicts
# parse_year(1997, yearstext)[:5]
yearinfo = []
for i in range(1992,2015):
    yearinfo.append(parse_year(i, yearstext))
```

```
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
```

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 (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 [5]:

```
parse_year(1997, yearstext)
```

```
1997
```

Out[5]:


```
[{'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']},
 {'band_singer': ['R. Kelly'],
  'ranking': 6,
  'song': ['I Believe I Can Fly'],
  'songurl': ['/wiki/I_Believe_I_Can_Fly'],
  'titletext': '"I Believe I Can Fly"',
  'url': ['/wiki/R._Kelly']},
 {'band_singer': ['En Vogue'],
  'ranking': 7,
  'song': ["Don't Let Go (Love)"],
  'songurl': ['/wiki/Don%27t_Let_Go_(Love)'],
  'titletext': '"Don\'t Let Go (Love)"',
  'url': ['/wiki/En_Vogue']},
 {'band_singer': ['Mark Morrison'],
  'ranking': 8,
  'song': ['Return of the Mack'],
  'songurl': ['/wiki/Return_of_the_Mack'],
  'titletext': '"Return of the Mack"',
  'url': ['/wiki/Mark_Morrison']},
 {'band_singer': ['LeAnn Rimes'],
  'ranking': 9,
  'song': ['How Do I Live'],
```

```
 'songurl': ['/wiki/How_Do_I_Live'],

 'titletext': '"How Do I Live"',
 'url': ['/wiki/LeAnn_Rimes']},
{'band_singer': ['Spice Girls'],
 'ranking': 10,
 'song': ['Wannabe'],
 'songurl': ['/wiki/Wannabe_(song)'],
 'titletext': '"Wannabe"',
 'url': ['/wiki/Spice_Girls']},
{'band_singer': ['Backstreet Boys'],
 'ranking': 11,
 'song': ['Quit Playing Games (With My Heart)'],
 'songurl': ['/wiki/Quit_Playing_Games_(With_My_Heart)'],
 'titletext': '"Quit Playing Games (With My Heart)"',
 'url': ['/wiki/Backstreet_Boys']},
{'band_singer': ['Hanson'],
 'ranking': 12,
 'song': ['MMMBop'],
 'songurl': ['/wiki/MMMBop'],
 'titletext': '"MMMBop"',
 'url': ['/wiki/Hanson_(band)']},
{'band_singer': ['Monica'],
 'ranking': 13,
 'song': ['For You I Will'],
 'songurl': ['/wiki/For_You_I_Will_(Monica_song)'],
 'titletext': '"For You I Will"',
 'url': ['/wiki/Monica_(singer)']},
{'band_singer': ['Usher'],
 'ranking': 14,
 'song': ['You Make Me Wanna...'],
 'songurl': ['/wiki/You_Make_Me_Wanna...'],
 'titletext': '"You Make Me Wanna..."',
 'url': ['/wiki/Usher_(entertainer)']},
{'band_singer': ['Meredith Brooks'],
 'ranking': 15,
 'song': ['Bitch'],
 'songurl': ['/wiki/Bitch_(Meredith_Brooks_song)'],
 'titletext': '"Bitch"',
 'url': ['/wiki/Meredith_Brooks']},
{'band_singer': ['Keith Sweat', 'Athena Cage'],
 'ranking': 16,
 'song': ['Nobody'],
 'songurl': ['/wiki/Nobody_(Keith_Sweat_song)'],
 'titletext': '"Nobody"',
 'url': ['/wiki/Keith_Sweat', '/wiki/Athena_Cage']},
{'band_singer': ['Third Eye Blind'],
 'ranking': 17,
 'song': ['Semi-Charmed Life'],
 'songurl': ['/wiki/Semi-Charmed_Life'],
 'titletext': '"Semi-Charmed Life"',
 'url': ['/wiki/Third_Eye_Blind']},
{'band_singer': ['Duncan Sheik'],
 'ranking': 18,
```

```
  'song': ['Barely Breathing'],

  'songurl': ['/wiki/Barely_Breathing'],
  'titletext': '"Barely Breathing"',
  'url': ['/wiki/Duncan_Sheik']},
 {'band_singer': ['Az Yet', 'Peter Cetera'],
  'ranking': 19,
  'song': ["Hard to Say I'm Sorry"],
  'songurl': ['/wiki/Hard_to_Say_I%27m_Sorry'],
  'titletext': '"Hard to Say I\'m Sorry"',
  'url': ['/wiki/Az_Yet', '/wiki/Peter_Cetera']},
 {'band_singer': ['The Notorious B.I.G.', 'Puff Daddy', 'Mase'],
  'ranking': 20,
  'song': ['Mo Money Mo Problems'],
  'songurl': ['/wiki/Mo_Money_Mo_Problems'],
  'titletext': '"Mo Money Mo Problems"',
  'url': ['/wiki/The_Notorious_B.I.G.', '/wiki/Sean_Combs', '/wiki/M
ase']},
 {'band_singer': ['The Verve Pipe'],
  'ranking': 21,
  'song': ['The Freshmen'],
  'songurl': ['/wiki/The_Freshmen_(song)'],
  'titletext': '"The Freshmen"',
  'url': ['/wiki/The_Verve_Pipe']},
 {'band_singer': ['Savage Garden'],
  'ranking': 22,
  'song': ['I Want You'],
  'songurl': ['/wiki/I_Want_You_(Savage_Garden_song)'],
  'titletext': '"I Want You"',
  'url': ['/wiki/Savage_Garden']},
{'band_singer': ['Blackstreet', 'Dr. Dre'],
 'ranking': 23,
 'song': ['No Diggity'],
 'songurl': ['/wiki/No_Diggity'],
 'titletext': '"No Diggity"',
 'url': ['/wiki/Blackstreet', '/wiki/Dr._Dre']},
{'band_singer': ['Rome'],
 'ranking': 24,
 'song': ['I Belong to You (Every Time I See Your Face)'],
 'songurl': ['/wiki/I_Belong_to_You_(Every_Time_I_See_Your_Face)'],
 'titletext': '"I Belong to You (Every Time I See Your Face)"',
 'url': ['/wiki/Rome_(musician)']},
{'band_singer': ['The Notorious B.I.G.'],
 'ranking': 25,
 'song': ['Hypnotize'],
 'songurl': ['/wiki/Hypnotize_(The_Notorious_B.I.G._song)'],
 'titletext': '"Hypnotize"',
 'url': ['/wiki/The_Notorious_B.I.G.']},
{'band_singer': ['Babyface'],
 'ranking': 26,
 'song': ['Every Time I Close My Eyes'],
 'songurl': ['/wiki/Every_Time_I_Close_My_Eyes'],
 'titletext': '"Every Time I Close My Eyes"',
 'url': ['/wiki/Babyface_(musician)']},
```

```
{'band_singer': ['Dru Hill'],

 'ranking': 27,
 'song': ['In My Bed'],
 'songurl': ['/wiki/In_My_Bed'],
 'titletext': '"In My Bed"',
 'url': ['/wiki/Dru_Hill']},
{'band_singer': ['Spice Girls'],
 'ranking': 28,
 'song': ["Say You'll Be There"],
 'songurl': ['/wiki/Say_You%27ll_Be_There'],
 'titletext': '"Say You\'ll Be There"',
 'url': ['/wiki/Spice_Girls']},
{'band_singer': ['Robyn'],
 'ranking': 29,
 'song': ['Do You Know (What It Takes)'],
 'songurl': ['/wiki/Do_You_Know_(What_It_Takes)'],
 'titletext': '"Do You Know (What It Takes)"',
 'url': ['/wiki/Robyn']},
{'band_singer': ['Boyz II Men'],
 'ranking': 30,
 'song': ['4 Seasons of Loneliness'],
 'songurl': ['/wiki/4_Seasons_of_Loneliness'],
 'titletext': '"4 Seasons of Loneliness"',
 'url': ['/wiki/Boyz_II_Men']},
{'band_singer': ['Changing Faces'],
 'ranking': 31,
 'song': ['G.H.E.T.T.O.U.T.'],
 'songurl': ['/wiki/G.H.E.T.T.O.U.T.'],
 'titletext': '"G.H.E.T.T.O.U.T."',
 'url': ['/wiki/Changing_Faces_(group)']},
{'band_singer': ['Mariah Carey'],
 'ranking': 32,
 'song': ['Honey'],
 'songurl': ['/wiki/Honey_(Mariah_Carey_song)'],
 'titletext': '"Honey"',
 'url': ['/wiki/Mariah_Carey']},
{'band_singer': ['Whitney Houston'],
 'ranking': 33,
 'song': ['I Believe in You and Me'],
 'songurl': ['/wiki/I_Believe_in_You_and_Me'],
 'titletext': '"I Believe in You and Me"',
 'url': ['/wiki/Whitney_Houston']},
{'band_singer': ['Freak Nasty'],
 'ranking': 34,
 'song': ["Da' Dip"],
 'songurl': ['/wiki/Da%27_Dip'],
 'titletext': '"Da\' Dip"',
 'url': ['/wiki/Freak_Nasty']},
{'band_singer': ['Spice Girls'],
 'ranking': 35,
 'song': ['2 Become 1'],
 'songurl': ['/wiki/2_Become_1'],
 'titletext': '"2 Become 1"',
```

```
 'url': ['/wiki/Spice_Girls']},

{'band_singer': ['Sister Hazel'],
 'ranking': 36,
 'song': ['All for You'],
 'songurl': ['/wiki/All_for_You_(Sister_Hazel_song)'],
 'titletext': '"All for You"',
 'url': ['/wiki/Sister_Hazel']},
{'band_singer': ['112'],
 'ranking': 37,
 'song': ['Cupid'],
 'songurl': ['/wiki/Cupid_(112_song)'],
 'titletext': '"Cupid"',
 'url': ['/wiki/112_(band)']},
{'band_singer': ['Paula Cole'],
 'ranking': 38,
 'song': ['Where Have All the Cowboys Gone?'],
 'songurl': ['/wiki/Where_Have_All_the_Cowboys_Gone%3F'],
 'titletext': '"Where Have All the Cowboys Gone?"',
 'url': ['/wiki/Paula_Cole']},
{'band_singer': ['Shawn Colvin'],
 'ranking': 39,
 'song': ['Sunny Came Home'],
 'songurl': ['/wiki/Sunny_Came_Home'],
 'titletext': '"Sunny Came Home"',
 'url': ['/wiki/Shawn_Colvin']},
{'band_singer': ['Tim McGraw', 'Faith Hill'],
 'ranking': 40,
 'song': ["It's Your Love"],
 'songurl': ['/wiki/It%27s_Your_Love'],
 'titletext': '"It\'s Your Love"',
 'url': ['/wiki/Tim_McGraw', '/wiki/Faith_Hill']},
{'band_singer': ['Gina G'],
 'ranking': 41,
 'song': ['Ooh Aah... Just a Little Bit'],
 'songurl': ['/wiki/Ooh_Aah..._Just_a_Little_Bit'],
 'titletext': '"Ooh Aah... Just a Little Bit"',
 'url': ['/wiki/Gina_G']},
{'band_singer': ['Merril Bainbridge'],
 'ranking': 42,
 'song': ['Mouth'],
 'songurl': ['/wiki/Mouth_(song)'],
 'titletext': '"Mouth"',
 'url': ['/wiki/Merril_Bainbridge']},
{'band_singer': ['Allure', '112'],
 'ranking': 43,
 'song': ['All Cried Out'],
 'songurl': ['/wiki/All_Cried_Out_(Lisa_Lisa_%26_Cult_Jam_song)'],
 'titletext': '"All Cried Out"',
 'url': ['/wiki/Allure_(group)', '/wiki/112_(band)']},
{'band_singer': ['New Edition'],
 'ranking': 44,
 'song': ["I'm Still in Love with You"],
 'songurl': ['/wiki/I%27m_Still_in_Love_with_You_(New_Edition_song)
```

```
']],

  'titletext': '"I\'m Still in Love with You"',
  'url': ['/wiki/New_Edition']},
 {'band_singer': ['98 Degrees'],
  'ranking': 45,
  'song': ['Invisible Man'],
  'songurl': ['/wiki/Invisible_Man_(song)'],
  'titletext': '"Invisible Man"',
  'url': ['/wiki/98_Degrees']},
 {'band_singer': ["Lil' Kim",
    'Da Brat',
    'Left Eye',
    'Missy Elliott',
    'Angie Martinez'],
  'ranking': 46,
  'song': ['Not Tonight'],
  'songurl': ['/wiki/Not_Tonight'],
  'titletext': '"Not Tonight"',
  'url': ['/wiki/Lil%27_Kim',
    '/wiki/Da_Brat',
    '/wiki/Lisa_Lopes',
    '/wiki/Missy_Elliott',
    '/wiki/Angie_Martinez']},
 {'band_singer': ['Bone Thugs-n-Harmony'],
  'ranking': 47,
  'song': ['Look into My Eyes'],
  'songurl': ['/wiki/Look_into_My_Eyes_(Bone_Thugs-n-Harmony_song)']
,
  'titletext': '"Look into My Eyes"',
  'url': ['/wiki/Bone_Thugs-n-Harmony']},
 {'band_singer': ['702'],
  'ranking': 48,
  'song': ['Get It Together'],
  'songurl': ['/wiki/Get_It_Together_(702_song)'],
  'titletext': '"Get It Together"',
  'url': ['/wiki/702_(band)']},
 {'band_singer': ['Celine Dion'],
  'ranking': 49,
  'song': ['All by Myself'],
  'songurl': ['/wiki/All_by_Myself#C.C3.A9line_Dion_version'],
  'titletext': '"All by Myself"',
  'url': ['/wiki/Celine_Dion']},
 {'band_singer': ['Celine Dion'],
  'ranking': 50,
  'song': ["It's All Coming Back to Me Now"],
  'songurl': ['/wiki/It%27s_All_Coming_Back_to_Me_Now'],
  'titletext': '"It\'s All Coming Back to Me Now"',
  'url': ['/wiki/Celine_Dion']},
 {'band_singer': ["Somethin' for the People", 'Trina & Tamara'],
  'ranking': 51,
  'song': ['My Love Is the Shhh!'],
  'songurl': ['/wiki/My_Love_Is_the_Shhh!'],
  'titletext': '"My Love Is the Shhh!"',
```

```
  'url': ['/wiki/Somethin%27_for_the_People', '/wiki/Trina_%26_Tamar

a']},
 {'band_singer': ['No Mercy'],
  'ranking': 52,
  'song': ['Where Do You Go'],
  'songurl': ['/wiki/Where_Do_You_Go_(No_Mercy_song)'],
  'titletext': '"Where Do You Go"',
  'url': ['/wiki/No_Mercy_(pop_band)']},
 {'band_singer': ['Barbra Streisand', 'Bryan Adams'],
  'ranking': 53,
  'song': ['I Finally Found Someone'],
  'songurl': ['/wiki/I_Finally_Found_Someone'],
  'titletext': '"I Finally Found Someone"',
  'url': ['/wiki/Barbra_Streisand', '/wiki/Bryan_Adams']},
 {'band_singer': ['Foxy Brown', 'Jay-Z'],
  'ranking': 54,
  'song': ["I'll Be"],
  'songurl': ['/wiki/I%27ll_Be_(Foxy_Brown_song)'],
  'titletext': '"I\'ll Be"',
  'url': ['/wiki/Foxy_Brown_(rapper)', '/wiki/Jay-Z']},
 {'band_singer': ['Sheryl Crow'],
  'ranking': 55,
  'song': ['If It Makes You Happy'],
  'songurl': ['/wiki/If_It_Makes_You_Happy'],
  'titletext': '"If It Makes You Happy"',
  'url': ['/wiki/Sheryl_Crow']},
 {'band_singer': ['Dru Hill'],
  'ranking': 56,
  'song': ['Never Make a Promise'],
  'songurl': ['/wiki/Never_Make_a_Promise'],
  'titletext': '"Never Make a Promise"',
  'url': ['/wiki/Dru_Hill']},
 {'band_singer': ['Journey'],
  'ranking': 57,
  'song': ['When You Love a Woman'],
  'songurl': ['/wiki/When_You_Love_a_Woman'],
  'titletext': '"When You Love a Woman"',
  'url': ['/wiki/Journey_(band)']},
 {'band_singer': ['Timbaland & Magoo', 'Missy Elliott', 'Aaliyah'],
  'ranking': 58,
  'song': ['Up Jumps da Boogie'],
  'songurl': ['/wiki/Up_Jumps_da_Boogie'],
  'titletext': '"Up Jumps da Boogie"',
  'url': ['/wiki/Timbaland_%26_Magoo',
   '/wiki/Missy_Elliott',
   '/wiki/Aaliyah']},
 {'band_singer': ['Toni Braxton'],
  'ranking': 59,
  'song': ["I Don't Want To", 'I Love Me Some Him'],
  'songurl': ['/wiki/I_Don%27t_Want_To', '/wiki/I_Love_Me_Some_Him']
,
  'titletext': '"I Don\'t Want To" / "I Love Me Some Him"',
  'url': ['/wiki/Toni_Braxton']},
```

```
{'band_singer': ['Sheryl Crow'],

 'ranking': 60,
 'song': ['Everyday Is a Winding Road'],
 'songurl': ['/wiki/Everyday_Is_a_Winding_Road'],
 'titletext': '"Everyday Is a Winding Road"',
 'url': ['/wiki/Sheryl_Crow']},
{'band_singer': ['MC Lyte'],
 'ranking': 61,
 'song': ['Cold Rock a Party'],
 'songurl': ['/wiki/Cold_Rock_a_Party'],
 'titletext': '"Cold Rock a Party"',
 'url': ['/wiki/MC_Lyte']},
{'band_singer': ['Ginuwine'],
 'ranking': 62,
 'song': ['Pony'],
 'songurl': ['/wiki/Pony_(Ginuwine_song)'],
 'titletext': '"Pony"',
 'url': ['/wiki/Ginuwine']},
{'band_singer': ['Sarah McLachlan'],
 'ranking': 63,
 'song': ['Building a Mystery'],
 'songurl': ['/wiki/Building_a_Mystery'],
 'titletext': '"Building a Mystery"',
 'url': ['/wiki/Sarah_McLachlan']},
{'band_singer': ['Donna Lewis'],
 'ranking': 64,
 'song': ['I Love You Always Forever'],
 'songurl': ['/wiki/I_Love_You_Always_Forever'],
 'titletext': '"I Love You Always Forever"',
 'url': ['/wiki/Donna_Lewis']},
{'band_singer': ['White Town'],
 'ranking': 65,
 'song': ['Your Woman'],
 'songurl': ['/wiki/Your_Woman'],
 'titletext': '"Your Woman"',
 'url': ['/wiki/White_Town']},
{'band_singer': ['Coolio'],
 'ranking': 66,
 'song': ['C U When U Get There'],
 'songurl': ['/wiki/C_U_When_U_Get_There'],
 'titletext': '"C U When U Get There"',
 'url': ['/wiki/Coolio']},
{'band_singer': ['Eric Clapton'],
 'ranking': 67,
 'song': ['Change the World'],
 'songurl': ['/wiki/Change_the_World'],
 'titletext': '"Change the World"',
 'url': ['/wiki/Eric_Clapton']},
{'band_singer': ['B-Rock and the Bizz'],
 'ranking': 68,
 'song': ['My Baby Daddy'],
 'songurl': ['/wiki/My_Baby_Daddy'],
 'titletext': '"My Baby Daddy"',
```

```
 'url': ['/wiki/B-Rock_and_the_Bizz']},

{'band_singer': ['Chumbawamba'],
 'ranking': 69,
 'song': ['Tubthumping'],
 'songurl': ['/wiki/Tubthumping'],
 'titletext': '"Tubthumping"',
 'url': ['/wiki/Chumbawamba']},
{'band_singer': ['R. Kelly'],
 'ranking': 70,
 'song': ['Gotham City'],
 'songurl': ['/wiki/Gotham_City_(song)'],
 'titletext': '"Gotham City"',
 'url': ['/wiki/R._Kelly']},
{'band_singer': ['Az Yet'],
 'ranking': 71,
 'song': ['Last Night'],
 'songurl': ['/wiki/Last_Night_(Az_Yet_song)'],
 'titletext': '"Last Night"',
 'url': ['/wiki/Az_Yet']},
{'band_singer': [],
 'ranking': 72,
 'song': ['The Jock Jam'],
 'songurl': ['/wiki/Jock_Jam'],
 'titletext': '"The Jock Jam"',
 'url': []},
{'band_singer': ['Heavy D'],
 'ranking': 73,
 'song': ['Big Daddy'],
 'songurl': ['/wiki/Big_Daddy_(song)'],
 'titletext': '"Big Daddy"',
 'url': ['/wiki/Heavy_D']},
{'band_singer': ['Total'],
 'ranking': 74,
 'song': ['What About Us'],
 'songurl': ['/wiki/What_About_Us%3F_(Total_song)'],
 'titletext': '"What About Us"',
 'url': ['/wiki/Total_(band)']},
{'band_singer': ['Scarface', '2Pac'],
 'ranking': 75,
 'song': ['Smile'],
 'songurl': ['/wiki/Smile_(Scarface_song)'],
 'titletext': '"Smile"',
 'url': ['/wiki/Scarface_(rapper)', '/wiki/Tupac_Shakur']},
{'band_singer': ['Montell Jordan'],
 'ranking': 76,
 'song': ["What's on Tonight"],
 'songurl': ['/wiki/What%27s_on_Tonight'],
 'titletext': '"What\'s on Tonight"',
 'url': ['/wiki/Montell_Jordan']},
{'band_singer': ['Bruce Springsteen'],
 'ranking': 77,
 'song': ['Secret Garden'],
 'songurl': ['/wiki/Secret_Garden_(Bruce_Springsteen_song)'],
```

```
 'titletext': '"Secret Garden"',

 'url': ['/wiki/Bruce_Springsteen']},
{'band_singer': ['Aaliyah'],
 'ranking': 78,
 'song': ['The One I Gave My Heart To'],
 'songurl': ['/wiki/The_One_I_Gave_My_Heart_To'],
 'titletext': '"The One I Gave My Heart To"',
 'url': ['/wiki/Aaliyah']},
{'band_singer': ['Seal'],
 'ranking': 79,
 'song': ['Fly Like an Eagle'],
 'songurl': ['/wiki/Fly_Like_an_Eagle_(song)#Cover_versions'],
 'titletext': '"Fly Like an Eagle"',
 'url': ['/wiki/Seal_(musician)']},
{'band_singer': ["Lil' Kim", 'Puff Daddy'],
 'ranking': 80,
 'song': ['No Time'],
 'songurl': ['/wiki/No_Time_(Lil%27_Kim_song)'],
 'titletext': '"No Time"',
 'url': ['/wiki/Lil%27_Kim', '/wiki/Sean_Combs']},
{'band_singer': ['Luscious Jackson'],
 'ranking': 81,
 'song': ['Naked Eye'],
 'songurl': ['/wiki/Naked_Eye_(song)'],
 'titletext': '"Naked Eye"',
 'url': ['/wiki/Luscious_Jackson']},
{'band_singer': ['Los del Río'],
 'ranking': 82,
 'song': ['Macarena'],
 'songurl': ['/wiki/Macarena_(song)'],
 'titletext': '"Macarena (Bayside Boys Mix)"',
 'url': ['/wiki/Los_del_R%C3%ADo']},
{'band_singer': ['Erykah Badu'],
 'ranking': 83,
 'song': ['On & On'],
 'songurl': ['/wiki/On_%26_On_(Erykah_Badu_song)'],
 'titletext': '"On & On"',
 'url': ['/wiki/Erykah_Badu']},
{'band_singer': ['Joe'],
 'ranking': 84,
 'song': '"Don\'t Wanna Be a Player"',
 'songurl': None,
 'titletext': '"Don\'t Wanna Be a Player"',
 'url': ['/wiki/Joe_(singer)']},
{'band_singer': ['Warren G'],
 'ranking': 85,
 'song': ['I Shot the Sheriff'],
 'songurl': ['/wiki/I_Shot_the_Sheriff_(Warren_G_song)'],
 'titletext': '"I Shot the Sheriff"',
 'url': ['/wiki/Warren_G']},
{'band_singer': ['Brian McKnight', 'Mase'],
 'ranking': 86,
 'song': ["You Should Be Mine (Don't Waste Your Time)"],
```

```
 'songurl': ['/wiki/You_Should_Be_Mine_(Don%27t_Waste_Your_Time)'],

 'titletext': '"You Should Be Mine (Don\'t Waste Your Time)"',
 'url': ['/wiki/Brian_McKnight', '/wiki/Mase']},
{'band_singer': ['Madonna'],
 'ranking': 87,
 'song': ["Don't Cry for Me Argentina"],
 'songurl': ['/wiki/Don%27t_Cry_for_Me_Argentina'],
 'titletext': '"Don\'t Cry for Me Argentina"',
 'url': ['/wiki/Madonna_(entertainer)']},
{'band_singer': ['SWV', 'Puff Daddy'],
 'ranking': 88,
 'song': '"Someone"',
 'songurl': None,
 'titletext': '"Someone"',
 'url': ['/wiki/SWV', '/wiki/Sean_Combs']},
{'band_singer': ['Michael Bolton'],
 'ranking': 89,
 'song': ['Go the Distance'],
 'songurl': ['/wiki/Go_the_Distance'],
 'titletext': '"Go the Distance"',
 'url': ['/wiki/Michael_Bolton']},
{'band_singer': ['Real McCoy'],
 'ranking': 90,
 'song': ['One More Time'],
 'songurl': ['/wiki/One_More_Time_(Real_McCoy_song)'],
 'titletext': '"One More Time"',
 'url': ['/wiki/Real_McCoy_(band)']},
{'band_singer': ['Next'],
 'ranking': 91,
 'song': '"Butta Love"',
 'songurl': None,
 'titletext': '"Butta Love"',
 'url': ['/wiki/Next_(group)']},
{'band_singer': ['Mr. President'],
 'ranking': 92,
 'song': ['Coco Jambo'],
 'songurl': ['/wiki/Coco_Jambo'],
 'titletext': '"Coco Jambo"',
 'url': ['/wiki/Mr._President_(band)']},
{'band_singer': ['Keith Sweat'],
 'ranking': 93,
 'song': ['Twisted'],
 'songurl': ['/wiki/Twisted_(Keith_Sweat_song)'],
 'titletext': '"Twisted"',
 'url': ['/wiki/Keith_Sweat']},
{'band_singer': ['Aqua'],
 'ranking': 94,
 'song': ['Barbie Girl'],
 'songurl': ['/wiki/Barbie_Girl'],
 'titletext': '"Barbie Girl"',
 'url': ['/wiki/Aqua_(band)']},
{'band_singer': ['The Cranberries'],
 'ranking': 95,
```

```
  'song': ["When You're Gone", 'Free to Decide'],

  'songurl': ['/wiki/When_You%27re_Gone_(The_Cranberries_song)',
   '/wiki/Free_to_Decide'],
  'titletext': '"When You\'re Gone" / "Free to Decide"',
  'url': ['/wiki/The_Cranberries']},
 {'band_singer': ['DJ Kool'],
  'ranking': 96,
  'song': ['Let Me Clear My Throat'],
  'songurl': ['/wiki/Let_Me_Clear_My_Throat'],
  'titletext': '"Let Me Clear My Throat"',
  'url': ['/wiki/DJ_Kool']},
 {'band_singer': ['The Blackout All-Stars'],
  'ranking': 97,
  'song': ['I Like It'],
  'songurl': ['/wiki/I_Like_It_Like_That_(Pete_Rodriguez_song)#The_B
lackout_All-Stars_version'],
  'titletext': '"I Like It"',
  'url': ['/wiki/The_Blackout_All-Stars']},
 {'band_singer': ['Toni Braxton'],
  'ranking': 98,
  'song': ["You're Makin' Me High", 'Let It Flow'],
  'songurl': ['/wiki/You%27re_Makin%27_Me_High', '/wiki/Let_It_Flow_
(song)'],
  'titletext': '"You\'re Makin\' Me High" / "Let It Flow"',
  'url': ['/wiki/Toni_Braxton']},
 {'band_singer': ['Madonna'],
  'ranking': 99,
  'song': ['You Must Love Me'],
  'songurl': ['/wiki/You_Must_Love_Me'],
  'titletext': '"You Must Love Me"',
  'url': ['/wiki/Madonna_(entertainer)']},
 {'band_singer': ['Ray J'],
  'ranking': 100,
  'song': ['Let It Go'],
  'songurl': ['/wiki/Let_It_Go_(Ray_J_song)'],
  'titletext': '"Let It Go"',
  'url': ['/wiki/Ray_J']}]
```


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.

DO NOT RERUN THE HTTP REQUESTS TO WIKIPEDIA WHEN SUBMITTING.

*We **DO NOT** need to see these JSON files in your submission!*

In [6]:

```
import json
```

In [ ]:

```
# DO NOT RERUN THIS CELL WHEN SUBMITTING
fd = open("data/yearinfo.json","w")
json.dump(yearinfo, fd)
fd.close()
del yearinfo
```

Now let's reload our JSON file into the yearinfo variable, just to be sure everything is working.

In [7]:

```
# RERUN WHEN SUBMITTING
# Another way to deal with files. Has the advantage of closing the file for you.
with open("data/yearinfo.json", "r") as fd:
    yearinfo = json.load(fd)
```

1.4 Construct a year-song-singer dataframe from the yearly information

Let's construct a dataframe `flatframe` from the `yearinfo`. The frame should be similar to the frame below. Each row of the frame represents a song, and carries with it the chief properties of year, song, singer, and ranking.

	year	band_singer	ranking	song	songurl	url
0	1992	Boyz II Men	1.0	End of the Road	/wiki/End_of_the_Road	/wiki/Boyz_II_Men
1	1993	Whitney Houston	1.0	I Will Always Love You	/wiki/I_Will_Always_Love_You#Whitney_Houston_v...	/wiki/Whitney_Houston
2	1994	Ace of Base	1.0	The Sign (song)	/wiki/The_Sign_(song)	/wiki/Ace_of_Base
3	1995	Coolio	1.0	Gangsta's Paradise	/wiki/Gangsta%27s_Paradise	/wiki/Coolio
4	1996	Los del Río	1.0	Macarena (song)	/wiki/Macarena_(song)	/wiki/Los_del_R%C3%ADo
5	1997	Elton John	1.0	Something About the Way You Look Tonight	/wiki/Something_About_the_Way_You_Look_Tonight	/wiki/Elton_John
6	1998	Next (group)	1.0	Too Close (Next song)	/wiki/Too_Close_(Next_song)	/wiki/Next_(group)
7	1999	Cher	1.0	Believe (Cher song)	/wiki/Believe_(Cher_song)	/wiki/Cher

To construct the dataframe, we'll need to iterate over the years and the singles per year. Notice how, above, the dataframe is ordered by ranking and then year. While the exact order is up to you, note that you will have to come up with a scheme to order the information.

Check that the dataframe has sensible data types. You will also likely find that the year field has become an "object" (Pandas treats strings as generic objects): this is due to the conversion to and back from JSON. Such conversions need special care. Fix any data type issues with `flatframe`. (See Pandas astype (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.astype.html) function.) We will use this `flatframe` in the next question.

(As an aside, we used the name `flatframe` to indicate that this dataframe is flattened from a hierarchical dictionary structure with the keys being the years.)

In [8]:

```
rows = []

for dict_list, y in zip(yearinfo, range(1992, 2015)):
    for d in dict_list:
        for i, singer in enumerate(d["band_singer"]):
            rows.append([y, d["ranking"], d["titletext"], d["songurl"], singer,
d["url"][i]])

flatframe = pd.DataFrame(rows, columns=["year", "ranking", "song", "songurl", "b
and_singer", "url"])
flatframe.sort_values(['ranking','year'], inplace=True)
flatframe.reset_index(inplace=True, drop=True)
flatframe

# ---------

# flatframes = []
# for d, y in zip(yearinfo, range(1992, 2015)):
#     cur_frame = pd.DataFrame(d)
#     cur_frame.insert(0, "year", y)
#     flatframes.append(cur_frame)

# flatframe = pd.concat(flatframes)
# del flatframe["song"]
# flatframe.rename(columns={'titletext': 'song'}, inplace=True)
# flatframe['songurl'] = flatframe['songurl'].str.get(0)

# left_frame = flatframe[["year", "ranking", "song", "songurl"]]
# right = flatframe[["song", "band_singer", "url", "year", "ranking"]].values.to
list()
# right_zipped = []
# for row in right:
#     for i, singer in enumerate(row[1]):
#         right_zipped.append( [row[0], singer, row[2][i], row[3], row[4]] )
# right_frame = pd.DataFrame(right_zipped, columns=["song", "band_singer", "url"
, "year", "ranking"])

# flatframe = pd.merge(left_frame, right_frame, on=["song", "year", "ranking"],
how="inner")
# flatframe.sort_values(['ranking','year'], inplace=True)
# flatframe.reset_index(inplace=True, drop=True)
# flatframe
```

Out[8]:

	year	ranking	song	songurl
0	1992	1	"End of the Road"	[/wiki/End_of_the_Road]

1	1993	1	"I Will Always Love You"	[/wiki/I_Will_Always_Love_You#Whitney_Houston_...
2	1994	1	"The Sign"	[/wiki/The_Sign_(song)]
3	1995	1	"Gangsta's Paradise"	[/wiki/Gangsta%27s_Paradise]
4	1995	1	"Gangsta's Paradise"	[/wiki/Gangsta%27s_Paradise]
5	1996	1	"Macarena (Bayside Boys Mix)"	[/wiki/Macarena_(song)]
6	1997	1	"Something About the Way You Look Tonight" / "...	[/wiki/Something_About_the_Way_You_Look_Tonigh...
7	1998	1	"Too Close"	[/wiki/Too_Close_(Next_song)]
8	1999	1	"Believe"	[/wiki/Believe_(Cher_song)]
9	2000	1	"Breathe"	[/wiki/Breathe_(Faith_Hill_song)]
10	2001	1	"Hanging by a Moment"	[/wiki/Hanging_by_a_Moment]
11	2002	1	"How You Remind Me"	[/wiki/How_You_Remind_Me]
12	2003	1	"In da Club"	[/wiki/In_da_Club]
13	2004	1	"Yeah!"	[/wiki/Yeah!_(Usher_song)]
14	2004	1	"Yeah!"	[/wiki/Yeah!_(Usher_song)]
15	2004	1	"Yeah!"	[/wiki/Yeah!_(Usher_song)]
16	2005	1	"We Belong Together"	[/wiki/We_Belong_Together_(Mariah_Carey_song)]
17	2006	1	"Bad Day"	[/wiki/Bad_Day_(Daniel_Powter_song)]
18	2007	1	"Irreplaceable"	[/wiki/Irreplaceable]
19	2008	1	"Low"	[/wiki/Low_(Flo_Rida_song)]
20	2008	1	"Low"	[/wiki/Low_(Flo_Rida_song)]
21	2009	1	"Boom Boom Pow"	[/wiki/Boom_Boom_Pow]
22	2010	1	"Tik Tok"	[/wiki/Tik_Tok]

23	2011	1	"Rolling in the Deep"	[/wiki/Rolling_in_the_Deep]
24	2012	1	"Somebody That I Used to Know"	[/wiki/Somebody_That_I_Used_to_Know]
25	2012	1	"Somebody That I Used to Know"	[/wiki/Somebody_That_I_Used_to_Know]
26	2013	1	"Thrift Shop"	[/wiki/Thrift_Shop]
27	2013	1	"Thrift Shop"	[/wiki/Thrift_Shop]
28	2013	1	"Thrift Shop"	[/wiki/Thrift_Shop]
29	2014	1	"Happy"	[/wiki/Happy_(Pharrell_Williams_song)]
...	...	...	...	...
2956	2014	99	"Somethin' Bad"	[/wiki/Somethin%27_Bad]
2957	1992	100	"I Can't Make You Love Me"	[/wiki/I_Can%27t_Make_You_Love_Me]
2958	1993	100	"Two Steps Behind"	[/wiki/Two_Steps_Behind]
2959	1994	100	"I Wanna Be Down"	[/wiki/I_Wanna_Be_Down]
2960	1995	100	"Can't Stop Lovin' You"	[/wiki/Can%27t_Stop_Lovin%27_You]
2961	1996	100	"Don't Cry"	[/wiki/Don%27t_Cry_(Seal_song)]
2962	1997	100	"Let It Go"	[/wiki/Let_It_Go_(Ray_J_song)]
2963	1998	100	"Westside"	[/wiki/Westside_(TQ_song)]
2964	1999	100	"(God Must Have Spent) A Little More Time on You"	[/wiki/(God_Must_Have_Spent)_A_Little_More_Tim...
2965	1999	100	"(God Must Have Spent) A Little More Time on You"	[/wiki/(God_Must_Have_Spent)_A_Little_More_Tim...
2966	2000	100	"Shackles (Praise You)"	[/wiki/Shackles_(Praise_You)]

2967	2001	100	"Hemorrhage (In My Hands)"	[/wiki/Hemorrhage_(In_My_Hands)]
2968	2002	100	"I Do!!"	[/wiki/I_Do!!]
2969	2003	100	"Say Yes"	[/wiki/Say_Yes_(Floetry_song)]
2970	2004	100	"I Go Back"	[/wiki/I_Go_Back]
2971	2005	100	"Give Me That"	[/wiki/Give_Me_That]
2972	2005	100	"Give Me That"	[/wiki/Give_Me_That]
2973	2006	100	"For You I Will (Confidence)"	[/wiki/For_You_I_Will_(Confidence)]
2974	2007	100	"Same Girl"	[/wiki/Same_Girl_(R._Kelly_and_Usher_song)]
2975	2007	100	"Same Girl"	[/wiki/Same_Girl_(R._Kelly_and_Usher_song)]
2976	2008	100	"She Got It"	[/wiki/She_Got_It]
2977	2008	100	"She Got It"	[/wiki/She_Got_It]
2978	2008	100	"She Got It"	[/wiki/She_Got_It]
2979	2009	100	"Never Say Never"	[/wiki/Never_Say_Never_(The_Fray_song)]
2980	2010	100	"Lover, Lover"	[/wiki/Lover,_Lover]
2981	2011	100	"My Last"	[/wiki/My_Last]
2982	2011	100	"My Last"	[/wiki/My_Last]
2983	2012	100	"Burn It Down"	[/wiki/Burn_It_Down_(Linkin_Park_song)]
2984	2013	100	"Still Into You"	[/wiki/Still_Into_You]
2985	2014	100	"Adore You"	[/wiki/Adore_You]

2986 rows × 6 columns

Who are the highest quality singers?

Here we show the highest quality singers and plot them on a bar chart.

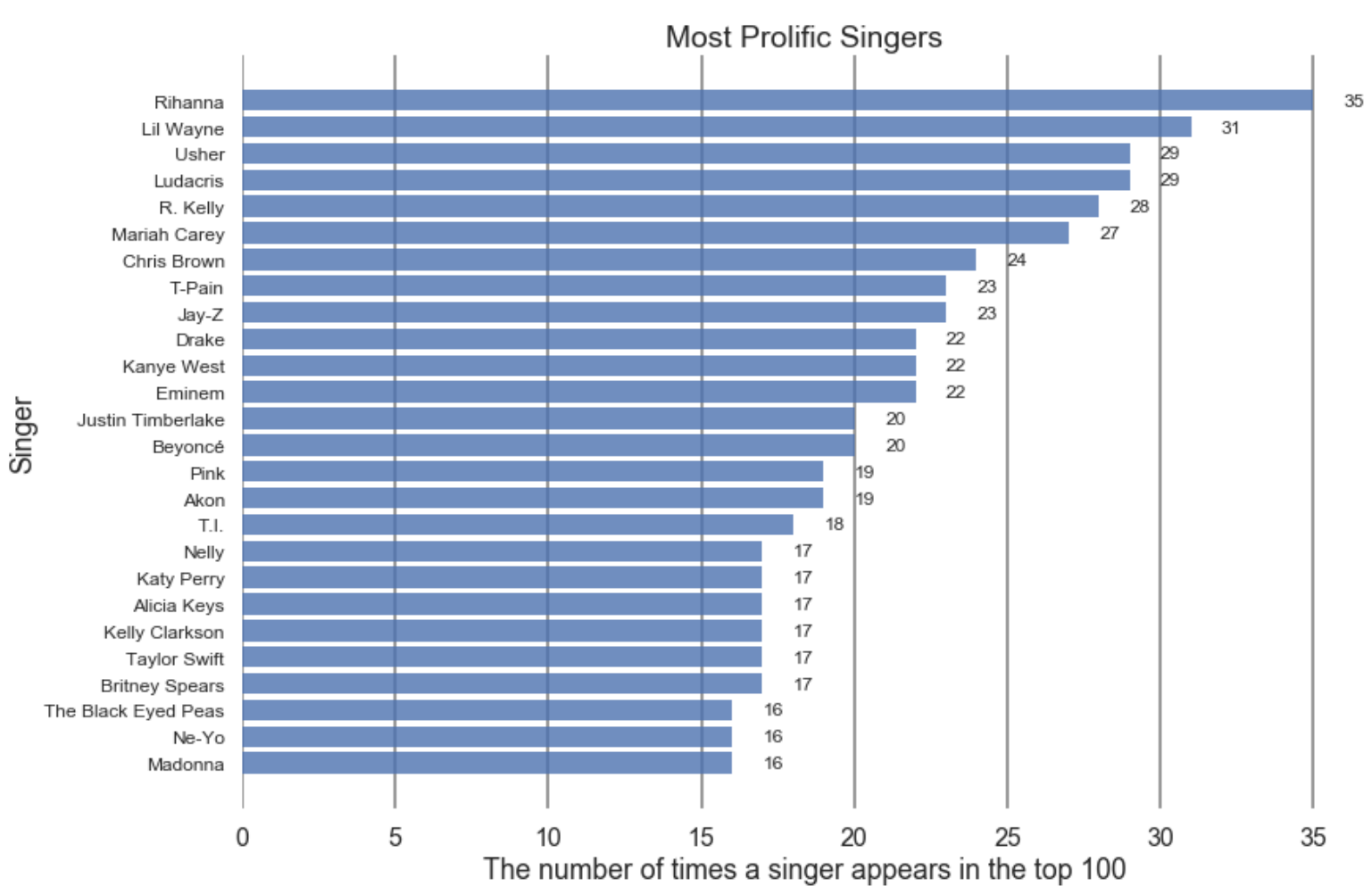
1.5 Find highest quality singers according to how prolific they are

What do we mean by highest quality? This is of course open to interpretation, but let's define "highest quality" here as the number of times a singer appears in the top 100 over this time period. If a singer appears twice in a year (for different songs), this is counted as two appearances, not one.

Make a bar-plot of the most prolific singers. Singers on this chart should have appeared at-least more than 15 times. (HINT: look at the docs for the pandas method `value_counts`.)

In [9]:

```
popular = flatframe['band_singer'].value_counts()
# print(popular)

popular_dict = dict(popular)
most_prolific = { key:value for key, value in popular_dict.items() if value > 15
}
# print(most_prolific)

prolific_sorted = sorted(most_prolific.items(), key=lambda x: x[1], reverse=Fals
e)
labels = [x[0] for x in prolific_sorted]
freqs = [x[1] for x in prolific_sorted]
y_coords = np.arange(len(most_prolific))

# freqs = most_prolific.values()
# labels = most_prolific.keys()

sns.set_style("whitegrid", {'axes.grid' : False})
ax = plt.gca()
plt.yticks(y_coords,labels, fontsize = 12)
plt.barh(y_coords,freqs, alpha=0.8)
plt.grid(axis = 'x', color ='grey', linestyle='-')
ax.tick_params(axis='both', which='both',length=0)
plt.xlabel("The number of times a singer appears in the top 100")
plt.ylabel("Singer")
plt.title("Most Prolific Singers")
for p, c, ch in zip(y_coords, labels, freqs):
    plt.annotate(str(ch), xy=(ch + 1, p), va='center', fontsize = 12)
sns.despine(left=True, bottom=True)
plt.show()

#------------Vertical Bar Graph Code-----------------------------
# freqs = most_prolific.values()
# x_coords = np.arange(len(most_prolific))
# labels = most_prolific.keys()
# plt.xticks(x_coords,labels, fontsize = 12, rotation = 90)
# plt.bar(x_coords,freqs)
# plt.ylabel("The number of times a singer appears in the top 100")
# plt.xlabel("Singer")
# plt.title("Most Prolific Singers")
# plt.show()
```



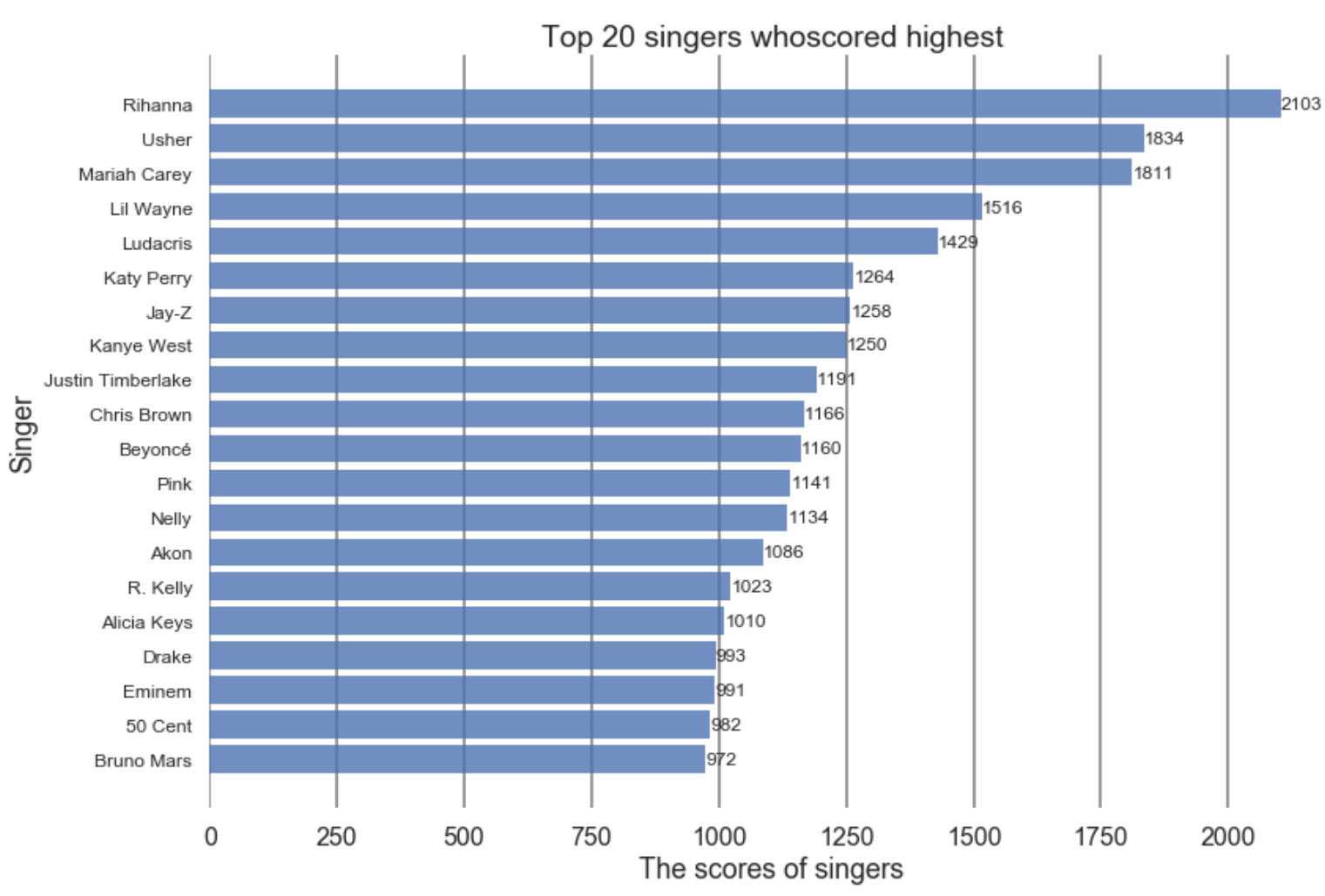
1.6 What if we used a different metric?

What we would like to capture is this: a singer should to be scored higher if the singer appears higher in the rankings. So we'd say that a singer who appeared once at a higher and once at a lower ranking is a "higher quality" singer than one who appeared twice at a lower ranking.

To do this, group all of a singers songs together and assign each song a score `101 - ranking`. Order the singers by their total score and make a bar chart for the top 20.

In [10]:

```
singer_score = flatframe.copy()
singer_score['score'] = 101 - singer_score['ranking']
g = singer_score.groupby(['band_singer'])['score'].sum()
g_dict = dict(g)
g_sorted = sorted(g_dict.items(), key=lambda x: x[1], reverse=True)
top20 = g_sorted[0:20]
# print(top20)


#--------------Vertical Bar Graph Code----------------------
# labels = [x[0] for x in top20]
# scores = [x[1] for x in top20]
# x_coords = np.arange(len(labels))
# plt.xticks(x_coords, labels, fontsize = 12, rotation = 90)
# plt.bar(x_coords,scores)
# plt.ylabel("The scores of singers")
# plt.xlabel("Singer")
# plt.title("Top 20 singers whoscored highest")
# plt.tight_layout()

top20_r = top20[::-1]
labels = [x[0] for x in top20_r]
scores = [x[1] for x in top20_r]
y_coords = np.arange(len(labels))

sns.set_style("whitegrid", {'axes.grid' : False})
ax = plt.gca()
plt.yticks(y_coords, labels, fontsize = 12)
plt.barh(y_coords,scores, alpha = 0.8)
plt.grid(axis = 'x', color ='gray', linestyle='-')
ax.tick_params(axis='both', which='both',length=0)
sns.despine(left=True, bottom=True)

plt.xlabel("The scores of singers")
plt.ylabel("Singer")
plt.title("Top 20 singers whoscored highest")
for p, c, ch in zip(y_coords, labels, scores):
    plt.annotate(str(ch), xy=(ch + 1, p), va='center', fontsize = 12)
plt.show()
```



1.7 Do you notice any major differences when you change the metric?

How have the singers at the top shifted places? Why do you think this happens?

In [11]:

```
print("Fist metric (# of appearances): \n", most_prolific)
print()
print("Second metric (Score): \n", top20)
```

```
Fist metric (# of appearances):
 {'Rihanna': 35, 'Lil Wayne': 31, 'Ludacris': 29, 'Usher': 29, 'R. K
elly': 28, 'Mariah Carey': 27, 'Chris Brown': 24, 'Jay-Z': 23, 'T-Pa
in': 23, 'Eminem': 22, 'Kanye West': 22, 'Drake': 22, 'Beyoncé': 20,
'Justin Timberlake': 20, 'Akon': 19, 'Pink': 19, 'T.I.': 18, 'Britne
y Spears': 17, 'Taylor Swift': 17, 'Kelly Clarkson': 17, 'Alicia Key
s': 17, 'Katy Perry': 17, 'Nelly': 17, 'Madonna': 16, 'Ne-Yo': 16, '
The Black Eyed Peas': 16}

Second metric (Score):
 [('Rihanna', 2103), ('Usher', 1834), ('Mariah Carey', 1811), ('Lil
Wayne', 1516), ('Ludacris', 1429), ('Katy Perry', 1264), ('Jay-Z', 1
258), ('Kanye West', 1250), ('Justin Timberlake', 1191), ('Chris Bro
wn', 1166), ('Beyoncé', 1160), ('Pink', 1141), ('Nelly', 1134), ('Ak
on', 1086), ('R. Kelly', 1023), ('Alicia Keys', 1010), ('Drake', 993
), ('Eminem', 991), ('50 Cent', 982), ('Bruno Mars', 972)]
```

Generally speaking, the two metrics give very similar results. However, we can still see some changes in ranking. For example, Mariah Carey hits Billboard 27 times and ranks in the 6th place using "appearance" metric; but she ranks in the third place using "score" metric, with a score very close to the second place. A more obvious change is Katy Perry, who ranks relatively low using "appearance" metric, but gets the 6th place with the "score" metric. This is probably because that although such singers don't have as many Billboard songs as others do, but every piece of their Billboard song is a very hot one, and hence each song attributes a very high score.

The counter examples are like R. Kelly, T.I and Britney Spears, for instance, whose ranks drop a lot if we use the second "score" metric. Although they have a lot of Billboard songs, but not many of them rank very high so their scores are not high either.

Q2. Scraping and Constructing: Information about Artists, Bands and Genres from Wikipedia

Our next job is to use those band/singer urls we collected under `flatframe.url` and get information about singers and/or bands.

Scrape information about artists from wikipedia

We wish to fetch information about the singers or groups for all the winning songs in a list of years.

Here we show a function that fetches information about a singer or group from their url on wikipedia. We create a cache object `urlcache` that will avoid redundant HTTP requests (e.g. an artist might have multiple singles on a single year, or be on the list over a span of years). Once we have fetched information about an artist, we don't need to do it again. The caching also helps if the network goes down, or the target website is having some problems. You simply need to run the `get_page` function below again, and the `urlcache` dictionary will continue to be filled.

If the request gets an HTTP return code different from 200, (such as a 404 not found or 500 Internal Server Error) the cells for that URL will have a value of 1; and if the request completely fails (e.g. no network connection) the cell will have a value of 2. This will allow you to analyse the failed requests.

Notice that we have wrapped the call in whats called *an exception block*. We try to make the request. If it fails entirely, or returns a HTTP code thats not 200, we set the status to 2 and 1 respectively.

In [12]:

```
urlcache={}
```

In [13]:

```
def get_page(url):
    # Check if URL has already been visited.
    if (url not in urlcache) or (urlcache[url]==1) or (urlcache[url]==2):
        time.sleep(1)
        # try/except blocks are used whenever the code could generate an excepti
on (e.g. division by zero).
        # In this case we don't know if the page really exists, or even if it do
es, if we'll be able to reach it.
        try:
            r = requests.get("http://en.wikipedia.org%s" % url)

            if r.status_code == 200:
                urlcache[url] = r.text
            else:
                urlcache[url] = 1
        except:
            urlcache[url] = 2
    return urlcache[url]
```

We sort the `flatframe` by year, ascending, first. Think why.

In [14]:

```
flatframe=flatframe.sort_values('year')
flatframe.tail()
```

Out[14]:

	year	ranking	song	songurl	band_sing
1489	2014	49	"This Is How We Roll"	[/wiki/This_Is_How_We_Roll]	Luke Bryan
2849	2014	95	"Partition"	[/wiki/Partition_(song)]	Beyoncé
731	2014	24	"Story of My Life"	[/wiki/Story_of_My_Life_(One_Direction_song)]	One Direction
1400	2014	46	"Roar"	[/wiki/Roar_(song)]	Katy Perry
2985	2014	100	"Adore You"	[/wiki/Adore_You]	Miley Cyrus

Pulling and saving the data

In [ ]:

```
# DO NOT RERUN THIS CELL WHEN SUBMITTING
# Here we are populating the url cache
# subsequent calls to this cell should be very fast, since Python won't
# need to fetch the page from the web server.
# NOTE this function will take quite some time to run (about 30 mins for me), si
nce we sleep 1 second before
# making a request. If you run it again it will be almost instantaneous, save re
quests that might have failed
# (you will need to run it again if requests fail..see cell below for how to tes
t this)
flatframe["url"].apply(get_page)
```

You may have to run this function again and again, in case there were network problems. Note that, because there is a "global" cache, it will take less time each time you run it. Also note that this function is designed to be run again and again: it attempts to make sure that there are no unresolved pages remaining. Let us make sure of this: *the sum below should be 0, and the boolean True.*

In [ ]:

```
# DO NOT RERUN THIS CELL WHEN SUBMITTING
print("Number of bad requests:",np.sum([(urlcache[k]==1) or (urlcache[k]==2) for
k in urlcache])) # no one or 0's)
print("Did we get all urls?", len(flatframe.url.unique())==len(urlcache)) # we g
ot all of the urls
```

Let's save the `urlcache` to disk, just in case we need it again.

In [ ]:

```
# DO NOT RERUN THIS CELL WHEN SUBMITTING
with open("data/artistinfo.json","w") as fd:
    json.dump(urlcache, fd)
del urlcache
```

In [15]:

```
# RERUN WHEN SUBMITTING
with open("data/artistinfo.json") as json_file:
    urlcache = json.load(json_file)
```

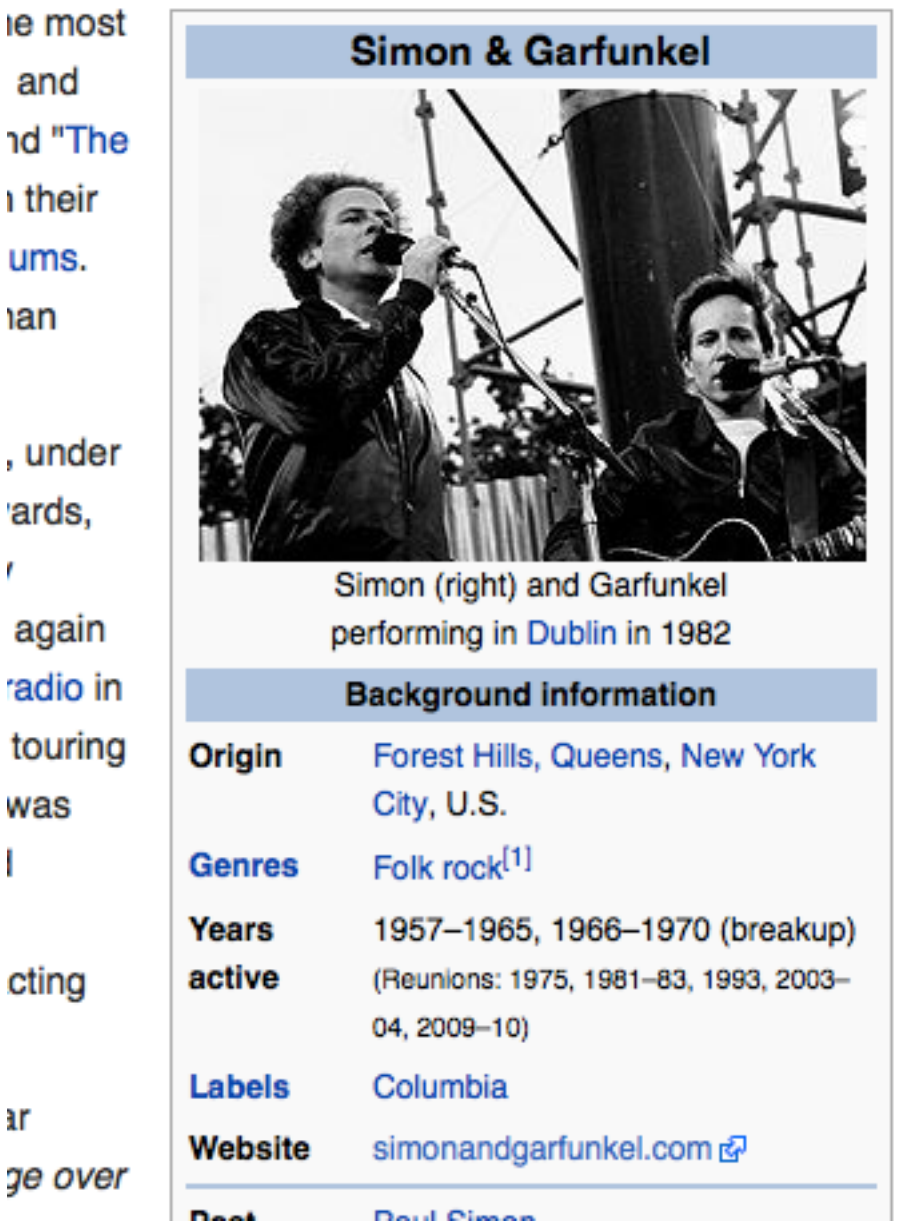
2.1 Extract information about singers and bands

From each page we collected about a singer or a band, extract the following information:

1. If the page has the text "Born" in the sidebar on the right, extract the element with the class `.bday`. If the page doesn't contain "Born", store `False`. Store either of these into the variable `born`. We want to analyze the artist's age.
2. If the text "Years active" is found, but no "born", assume a band. Store into the variable `ya` the value of the next table cell corresponding to this, or `False` if the text is not found.

Put this all into a function `singer_band_info` which takes the singer/band url as argument and returns a dictionary `dict(url=url, born=born, ya=ya)`.

The information can be found on the sidebar on each such wikipedia page, as the example here shows:



.

Write the function `singer_band_info` according to the following specification:

In [16]:

```
"""
Function
--------
singer_band_info

Inputs
------
url: the url
page_text: the text associated with the url

Returns
-------
A dictionary with the following data:
    url: copy the input argument url into this value
    born: the artist's birthday
    ya: years active variable

Notes
-----
See description above. Also note that some of the genres urls might require a
bit of care and special handling.
"""

def singer_band_info(url, page_text):
    info_dic = dict()
    info_dic["url"] = url
    soup = BeautifulSoup(page_text, 'html.parser')
    if not soup.find(text="Years active") is None:
        info_dic["ya"] = soup.find(text="Years active").parent.parent.find_next(
"td").text
    else:
        info_dic["ya"] = False

    if not soup.find(text="Born") is None:
        td = soup.find(text="Born").parent.parent.find("td")
        if not td.find("span", {"class": "bday"}) is None:
            info_dic["born"] = td.find("span", {"class": "bday"}).text
        else:
            info_dic["born"] = False
    else:
        info_dic["born"] = False

    return info_dic


# singer_band_info("/wiki/Jeremy_Jordan_(singer)", urlcache["/wiki/Jeremy_Jordan
_(singer)"])
```

2.2 Merging this information in

Iterate over the items in the singer-group dictionary cache `urlcache`, run the above function, and create a dataframe from there with columns `url`, `born`, and `ya`. Merge this dataframe on the `url` key with `flatframe`, creating a rather wide dataframe that we shall call `largedf`. It should look something like this:

	year	band_singer	ranking	song	songurl	url	born	ya
0	1992	Boyz II Men	1.0	End of the Road	/wiki/End_of_the_Road	/wiki/Boyz_II_Men	False	1985-prese
1	1992	Boyz II Men	37.0	It's So Hard to Say Goodbye to Yesterday	/wiki/It%27s_So_Hard_to_Say_Goodbye_to_Yesterday	/wiki/Boyz_II_Men	False	1985-prese
2	1992	Boyz II Men	84.0	Uhh Ahh	/wiki/Uhh_Ahh	/wiki/Boyz_II_Men	False	1985-prese
3	1993	Boyz II Men	12.0	In the Still of the Night (1956 song)	/wiki/In_the_Still_of_the_Night_(1956_song)#Bo...	/wiki/Boyz_II_Men	False	1985-prese
4	1994	Boyz II Men	3.0	I'll Make Love to You	/wiki/I%27ll_Make_Love_to_You	/wiki/Boyz_II_Men	False	1985-prese

Notice how the `born` and `ya` and `url` are repeated every time a different song from a given band is represented in a row.

In [17]:

```
# ----------Test--------------------
# for u in urlcache:
#     try:
#         info_dic = singer_band_info(u,urlcache[u])
#     except:
#         print(u)

info_dic = dict()
bio_rows = []
for u in urlcache:
    info_dic = singer_band_info(u, urlcache[u])
    bio_rows.append([info_dic["url"], info_dic["born"], info_dic["ya"]])

# print(bio_rows)
```

In [18]:

```
bio_df = pd.DataFrame(bio_rows, columns=["url", "born", "ya"])
bio_df
largedf = pd.merge(flatframe, bio_df, on = "url", how = "inner")
# largedf.sort_values(['band_singer','year','ranking'], inplace=True)
largedf
```

Out[18]:

	year	ranking	song	songurl	ba
0	1992	1	"End of the Road"	[/wiki/End_of_the_Road]	Bo
1	1992	37	"It's So Hard to Say Goodbye to Yesterday"	[/wiki/It%27s_So_Hard_to_Say_Goodbye_to_Yester...	Bo
2	1992	84	"Uhh Ahh"	[/wiki/Uhh_Ahh]	Bo
3	1993	12	"In the Still of the Nite"	[/wiki/In_the_Still_of_the_Night_(1956_song)#B...	Bo
4	1994	3	"I'll Make Love to You"	[/wiki/I%27ll_Make_Love_to_You]	Bo
5	1995	5	"On Bended Knee"	[/wiki/On_Bended_Knee]	Bo
6	1995	12	"Water Runs Dry"	[/wiki/Water_Runs_Dry]	Bo
7	1995	50	"I'll Make Love to You"	[/wiki/I%27ll_Make_Love_to_You]	Bo
8	1995	94	"Thank You"	[/wiki/Thank_You_(Boyz_II_Men_song)]	Bo
9	1996	2	"One Sweet Day"	[/wiki/One_Sweet_Day]	Bo
10	1997	30	"4 Seasons of Loneliness"	[/wiki/4_Seasons_of_Loneliness]	Bo
11	1998	96	"4 Seasons of	[/wiki/4_Seasons_of_Loneliness]	Bo

			Loneliness"		
12	1998	30	"A Song for Mama"	[/wiki/A_Song_for_Mama]	B
13	1992	41	"The Best Things in Life Are Free"	[/wiki/The_Best_Things_in_Life_Are_Free]	Lu Va
14	1994	56	"Endless Love"	[/wiki/Endless_Love_(song)#Luther_Vandross_and...	Lu Va
15	1992	69	"Wildside"	[/wiki/Wildside_(Marky_Mark_and_the_Funky_Bunc...	M ar Fu
16	1992	40	"Tennessee"	[/wiki/Tennessee_(Arrested_Development_song)]	Ar D
17	1992	67	"People Everyday"	[/wiki/People_Everyday]	Ar D
18	1993	31	"Mr. Wendal"	[/wiki/Mr._Wendal]	Ar D
19	1992	70	"Do I Have to Say the Words?"	[/wiki/Do_I_Have_to_Say_the_Words%3F]	Br Ac
20	1992	74	"Thought I'd Died and Gone to Heaven"	[/wiki/Thought_I%27d_Died_and_Gone_to_Heaven]	Br Ac
21	1994	27	"Please Forgive Me"	[/wiki/Please_Forgive_Me]	Br Ac
22	1994	8	"All for Love"	[/wiki/All_for_Love_(song)]	Br Ac
23	1995	16	"Have You Ever Really Loved a Woman?"	[/wiki/Have_You_Ever_Really_Loved_a_Woman%3F]	Br Ac
24	1997	53	"I Finally Found Someone"	[/wiki/I_Finally_Found_Someone]	Br Ac

25	1992	39	"Bohemian Rhapsody"	[/wiki/Bohemian_Rhapsody]	Q
26	1992	71	"Friday I'm in Love"	[/wiki/Friday_I%27m_in_Love]	Th
27	1992	38	"Move This"	[/wiki/Move_This]	Ya
28	1992	72	"Everything About You"	[/wiki/Everything_About_You_(Ugly_Kid_Joe_song)]	Ug
29	1993	61	"Cat's in the Cradle"	[/wiki/Cat%27s_in_the_Cradle]	Ug
...	...	...	...	...	...
2955	2014	90	"23"	[/wiki/23_(song)]	Ju
2956	2014	72	"Lifestyle"	[/wiki/Lifestyle_(song)]	Yo
2957	2014	12	"Pompeii"	[/wiki/Pompeii_(Bastille_song)]	Ba
2958	2014	72	"Lifestyle"	[/wiki/Lifestyle_(song)]	Ri
2959	2014	30	"Loyal"	[/wiki/Loyal_(Chris_Brown_song)]	Fr M
2960	2014	93	"She Looks So Perfect"	[/wiki/She_Looks_So_Perfect]	5 Su
2961	2014	86	"Amnesia"	[/wiki/Amnesia_(5_Seconds_of_Summer_song)]	5 Su
2962	2014	41	"Rather Be"	[/wiki/Rather_Be_(Clean_Bandit_song)]	Cl
2963	2014	41	"Rather Be"	[/wiki/Rather_Be_(Clean_Bandit_song)]	Je
2964	2014	42	"Don't Tell 'Em"	[/wiki/Don%27t_Tell_%27Em]	YC
2965	2014	58	"My Hitta"	[/wiki/My_Nigga]	YC
2966	2014	28	"Latch"	[/wiki/Latch_(song)]	Di

2967	2014	88	"Come with Me Now"	[/wiki/Come_with_Me_Now]	K(
2968	2014	76	"Leave the Night On"	[/wiki/Leave_the_Night_On]	Sa
2969	2014	43	"Show Me"	[/wiki/Show_Me_(Kid_Ink_song)]	Ki
2970	2014	31	"Best Day of My Life"	[/wiki/Best_Day_of_My_Life]	Ar Au
2971	2014	7	"Rude"	[/wiki/Rude_(song)]	M
2972	2014	32	"Habits (Stay High)"	[/wiki/Habits_(Stay_High)]	To
2973	2014	92	"White Walls"	[/wiki/White_Walls]	M & Le
2974	2014	92	"White Walls"	[/wiki/White_Walls]	H(
2975	2014	90	"23"	[/wiki/23_(song)]	M M
2976	2014	38	"Bailando"	[/wiki/Bailando_(Enrique_Iglesias_song)]	D(Bu
2977	2014	38	"Bailando"	[/wiki/Bailando_(Enrique_Iglesias_song)]	G(Zo
2978	2014	96	"Studio"	[/wiki/Studio_(song)]	B(C(
2979	2014	54	"Hot Boy"	[/wiki/Hot_Nigga]	B(Sh
2980	2014	21	"Let It Go"	[/wiki/Let_It_Go_(Disney_song)]	Id
2981	2014	50	"Classic"	[/wiki/Classic_(MKTO_song)]	M
2982	2014	26	"Black Widow"	[/wiki/Black_Widow_(song)]	Ri
2983	2014	8	"All About That Bass"	[/wiki/All_About_That_Bass]	M Tr
2984	2014	48	"The Man"	[/wiki/The_Man_(Aloe_Blacc_song)]	Al

2985 rows × 8 columns

2.3 What is the age at which singers achieve their top ranking?

Plot a histogram of the age at which singers achieve their top ranking. What conclusions can you draw from this distribution of ages?

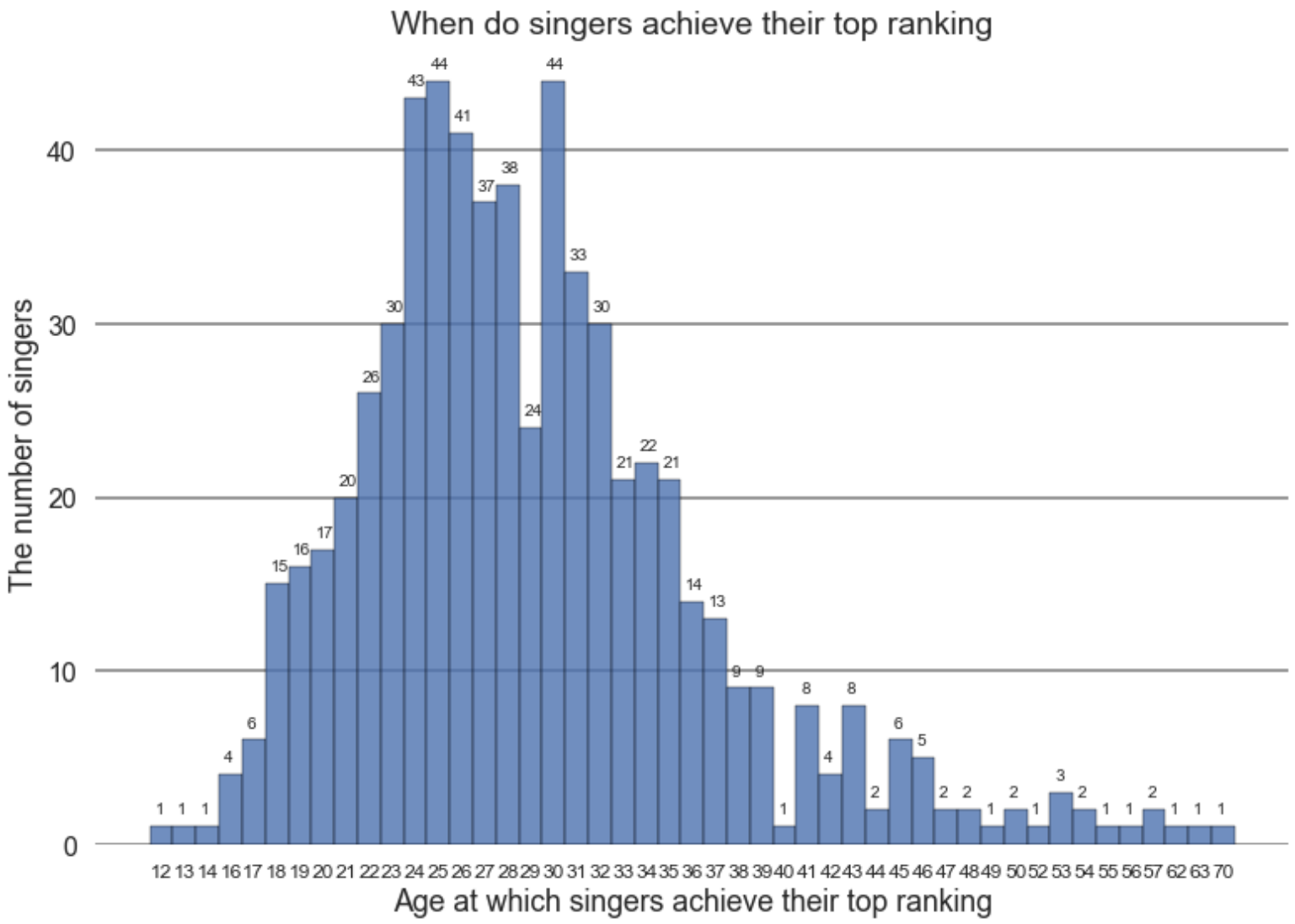
HINT: You will need to do some manipulation of the `born` column, and find the song for which a band or an artist achieves their top ranking. You will then need to put these rows together into another dataframe or array to make the plot.

In [19]:

```
age_df = largedf.copy()
age_df.drop(age_df[age_df.born == False].index, inplace=True)
age_df["birth_year"] = age_df["born"].str.extract('(\d+)').astype(int)

# age_df["birth_year"] = age_df["born"].str.get(0)+\
#                        age_df["born"].str.get(1)+age_df["born"].str.get(2)+age
_df["born"].str.get(3)
# age_df["birth_year"] = (age_df["birth_year"]).astype(int)

age_df["age_then"] = age_df["year"] - age_df["birth_year"]
```

```
age_df.reset_index(inplace=True, drop=True)
grouped = age_df.groupby(["band_singer"])
age = grouped.apply(lambda x: x.sort_values("ranking").head(1))

age_list = age["age_then"].tolist()
age_count = {x:age_list.count(x) for x in age_list}
age_sorted = sorted(age_count.items(), key=lambda x: x[0])
age_sorted

labels = [x[0] for x in age_sorted]
freqs = [x[1] for x in age_sorted]
x_coords = np.arange(len(labels))
plt.xticks(x_coords, labels, fontsize = 12)
plt.bar(x_coords,freqs, width=1.0, edgecolor="black", alpha=0.8)
plt.ylabel("The number of singers")
plt.xlabel("Age at which singers achieve their top ranking")
plt.title("When do singers achieve their top ranking")
plt.grid(axis = 'y', color ='gray', linestyle='-')
sns.despine(left=True, bottom=True)
for p, ch in zip(x_coords, freqs):
    plt.annotate(str(ch), xy=(p-0.3,ch+1), va='center', fontsize = 10)
plt.show()
```

```
/Applications/anaconda/lib/python3.6/site-packages/ipykernel_launche
r.py:3: FutureWarning: currently extract(expand=None) means expand=F
alse (return Index/Series/DataFrame) but in a future version of pand
as this will be changed to expand=True (return DataFrame)
  This is separate from the ipykernel package so we can avoid doing
imports until
```



From the histogram we can see the peak age range that most singers achieve their top ranking is about 24 to 30. Generally speaking, most singers achieve their top ranking between 18 and late thirties. This makes sense because musical performance requires not only talent, but also accumulation of experience, and that's why no many singers achieve their top before 18. On the other hand, when a singer gets old, it is very likely that he/she cannot perform as good as when he/she was young. That explains why not many singers achieve top ranking especially after early forties.

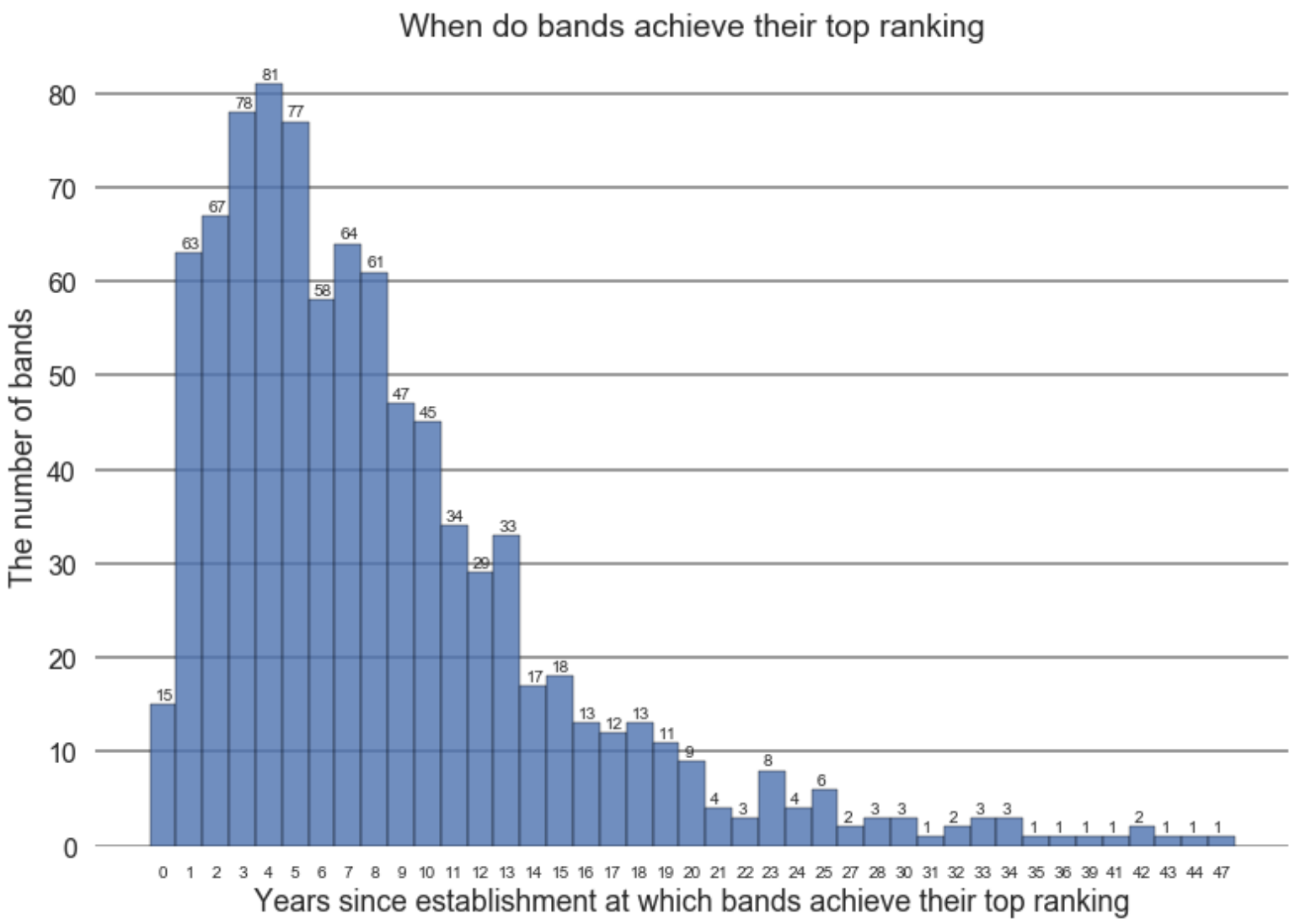
2.4 At what year since inception do bands reach their top rankings?

Make a similar calculation to plot a histogram of the years since inception at which bands reach their top ranking. What conclusions can you draw?

In [20]:

```
band_df = largedf.copy()
band_df.drop(band_df[band_df.ya == False].index, inplace=True)
band_df["inception_year"] = band_df["ya"].str.extract('(\d+)').astype(int)

# band_df["inception_year"] = band_df["ya"].str.get(0)+\
#                             band_df["ya"].str.get(1)+band_df["ya"].str.get(2)+
band_df["ya"].str.get(3)
# band_df.drop(band_df[band_df.inception_year == "Late"].index, inplace=True)
# band_df.drop(band_df[band_df.inception_year == "Musi"].index, inplace=True)

# band_df["inception_year"] = (band_df["inception_year"]).astype(int)
band_df["years_of_establishment"] = band_df["year"] - band_df["inception_year"]



band_df.reset_index(inplace=True, drop=True)
b_grouped = band_df.groupby(["band_singer"])
band = b_grouped.apply(lambda x: x.sort_values("ranking").head(1))

band_list = band["years_of_establishment"].tolist()
band_count = {x:band_list.count(x) for x in band_list}
band_sorted = sorted(band_count.items(), key=lambda x: x[0])
band_sorted

labels = [x[0] for x in band_sorted]
freqs = [x[1] for x in band_sorted]
x_coords = np.arange(len(labels))
plt.xticks(x_coords, labels, fontsize=10)
plt.bar(x_coords, freqs, width=1.0, edgecolor="black", alpha=0.8)
plt.ylabel("The number of bands")
plt.xlabel("Years since establishment at which bands achieve their top ranking")
plt.title("When do bands achieve their top ranking")
plt.grid(axis = 'y', color ='gray', linestyle='-')
sns.despine(left=True, bottom=True)
for p, ch in zip(x_coords, freqs):
    plt.annotate(str(ch), xy=(p-0.3,ch+1), va='center', fontsize = 10)
plt.show()
```

```
/Applications/anaconda/lib/python3.6/site-packages/ipykernel_launche
r.py:3: FutureWarning: currently extract(expand=None) means expand=F
alse (return Index/Series/DataFrame) but in a future version of pand
as this will be changed to expand=True (return DataFrame)
  This is separate from the ipykernel package so we can avoid doing
imports until
```



From the histogram, we can see that the peak period that bands achieve their top ranking is between 1 to 8 years after their establishment. Generally speaking, most bands achieve their top ranking between 1 to 13 years. This makes sense because it is very natural that bands break up after several years. From the histogram, we can also see that some bands get their top ranking even decades after establishment. This could happen for reasons such as they got reunion several years after break-up.

In [ ]: