Problem Chosen	2021 MCM/ICM Summary Sheet	Team Control Number
D		2112323

The Influence Of Music

Summary

In this paper, we explore a way to understand and measure the influence of previously produced music on new music and musical artists.

For question 1, a directed network is set up to capture music influence. In this paper, we used PageRank algorithm to compute node influence. Here we develop an influence factor parameter described by the PR value of each node that captures '*music influence*' in this network. Artists with a higher influence factor have more influence.

For question 2, we choose Mahalanobis distance to compute the similarity based on that the features are correlated. The main idea is to rotate the variables according to the principal components, make the dimensions independent of each other, and then standardize them. Then, use Silhouette coefficient to prove that artists within genre are more similar than artists between genres.

Keywords: Influence of Music; Directed Network; Similarity; PageRank

Contents

1 Introduction

In the historical process of human development, music is an indispensable part. Artists will be influenced by many factors when they create music.We hope to understand and measure the influence of previously produced music on new music and musical artists.

Different people have put forward different measurement methods. But the change of music is not only affected by a single factor, it may be due to the cooperative efforts of artists, social or political events, etc. In addition, the interaction between artists will also change the music. Meanwhile, by considering networks of songs and their musical characteristics, we can study the influence of artists on each other, and also know the evolution of music in the process of social development.

We will develop a model to measure the influence of music by answering the following questions raised the ICM Society:

- Create a (multiple) directed network(s) of musical influence, where influencers are connected to followers and Develop parameters that capture 'music influence' in this network. Explore a subset of musical influence by creating a subnetwork of your directed influencer network. Describe this subnetwork. What do your 'music influence' measures reveal in this subnetwork?
- Try to develop measures of music similarity. Using this measure, are artists within genre more similar than artists between genres?
- Indicate whether the similarity data, as reported in the data_influence data set, suggest that the identified influencers in fact influence the respective artists. Do the influencers actually affect the music created by the followers? Are some music characteristics more contagious than others, or do they all have similar roles in influencing a particular artist's music?
- Identify if there are characteristics that might signify revolutions (major leaps) in musical evolution from these data? What artists represent revolutionaries (influencers of major change) in your network?
- Analyze the influence processes of musical evolution that occurred over time in one genre. Can your team identify indicators that reveal the dynamic influencers, and explain how the genre(s) or artist(s) changed over time?
- How does your work express information about cultural influence of music in time or circumstances? Alternatively, how can the effects of social, political or technological changes (such as the internet) be identified within the network?

2 Influence Ranking

2.1 Establishment of Directed Influence Network

A directed network is set up to capture music influence using links from influencers to respective followers both of which are represented by nodes. with different

colors representing different genres as shown in Figure 1. We tried several graph layout algorithms such as classical force-directed layouts, and it turned out that the DrL graph layout algorithm had the best result which we choose to apply in this article.

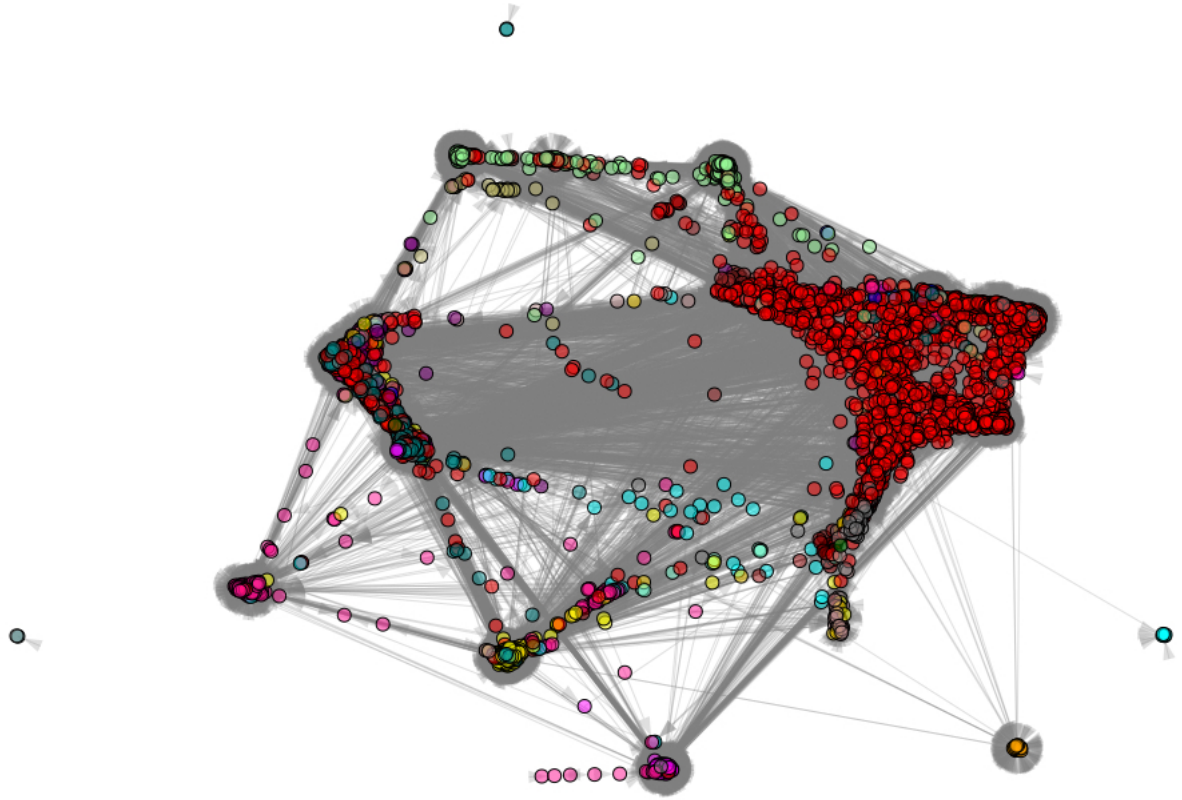

Figure 1: Primitive Directed Influence Network

2.2 Influence Ranking Using PageRank

Researchers have proposed several effective approaches to rank the importance of nodes in network research area. An intuitive way is to compute centralities of nodes such as Degree centrality and Betweenness centrality. However, centrality has limitations, for example, it ignores the position of a node and the influence of its neighbor nodes. While Google proposed PageRank algorithm applied by Google search engine, which borrowed the primary idea of centrality and further developed. In this paper, we used PageRank algorithm to compute node influence.

The PageRank algorithm uses the output probability distribution to represent the probability of someone randomly clicking on a web page. The probability is evenly distributed to each page in the beginning. Then, iterative computations are required to adjust PageRank values to converge to theoretical value. The PageRank value for any page u can be expressed as:

$$PR(p_i) = \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)} \tag{1}$$

where $PR(p_i)$ is the PR value of web page p_i; $M(p_i)$ represents the collection of all the pages linked to page p_i; $L(p_j)$ represents the number of outbound links of web page p_j.

According to the PageRank theory, a surfer who is clicking on links at random will eventually stop. The probability of proceeding at any step is a damping factor d, it is generally assumed that the damping factor will be set around 0.85. If a sink page is arrived at, any other pages can be linked to with equal probability. Thus, the scores of sink pages with no outbound links are divided evenly among all other pages. So, the equation is as follows:

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)} \tag{2}$$

where N is the total number of pages.

With regard to the question discussed in this article, we apply equation(2) to *influence_data*. Here $PR(p_i)$ represents the degree of influence of artist p_i, $M(p_i)$ represents the collection of all the artists linked from artist p_i, and $L(p_j)$ represents the number of inbound links of artist p_j. N should represent the number of the artists originally, but given that artists were only likely to have an impact on those who became active later on, we put the number of artists whose active start year is earlier than artist p_i into N. It should also be noted that all of the links are reversed compared to the orientation of the corresponding links in PageRank because the probability of browsing a web page depends on the links to it while the influence of an artist is exerted on others.

2.3 Influence Factor

On the basis of Figure 1, the diameter of nodes in the network is proportional to the PR value of each node which represents the influence of artists as shown in Figure 2. Here we develop an influence factor parameter that captures '*music influence*' in this network. Artists with a higher influence factor had an impact on others more profoundly. We extract top 10 artists with the highest influence factor as shown in Chart 1 which can be verified by historical facts of music.

Rank	ID	Name	Main Genre	Active Start	Page Rank
1	532957	Cab Calloway	Jazz	1930	0.01794
2	79016	Billie Holiday	Vocal	1930	0.01648
3	259529	Lester Young	Jazz	1930	0.01382
4	287604	Louis Jordan	Jazz	1930	0,01212
5	754032	The Beatles	Pop/Rock	1960	0.00941
6	3829	T-Bone Walker	Blues	1930	0.00896
7	13511	Sister Rosetta Tharpe	Religious	1930	0.00780
8	403120	The Mills Brothers	Vocal	1930	0.00673
9	898331	Mississippi Fred McDowell	Blues	1950	0.00608
10	66915	Bob Dylan	Pop/Rock	1960	0.00606

Table 1: Top 10 Influencer

In order to observe the influence of artists on each other explicitly and to reveal the meaning of influence factor, the genre of Latin was extracted due to its sparsity as a case study as shown in Figure 3. There are two nodes with relatively large diameters with more outbound links pointing to others, that is has a broader impact on others.

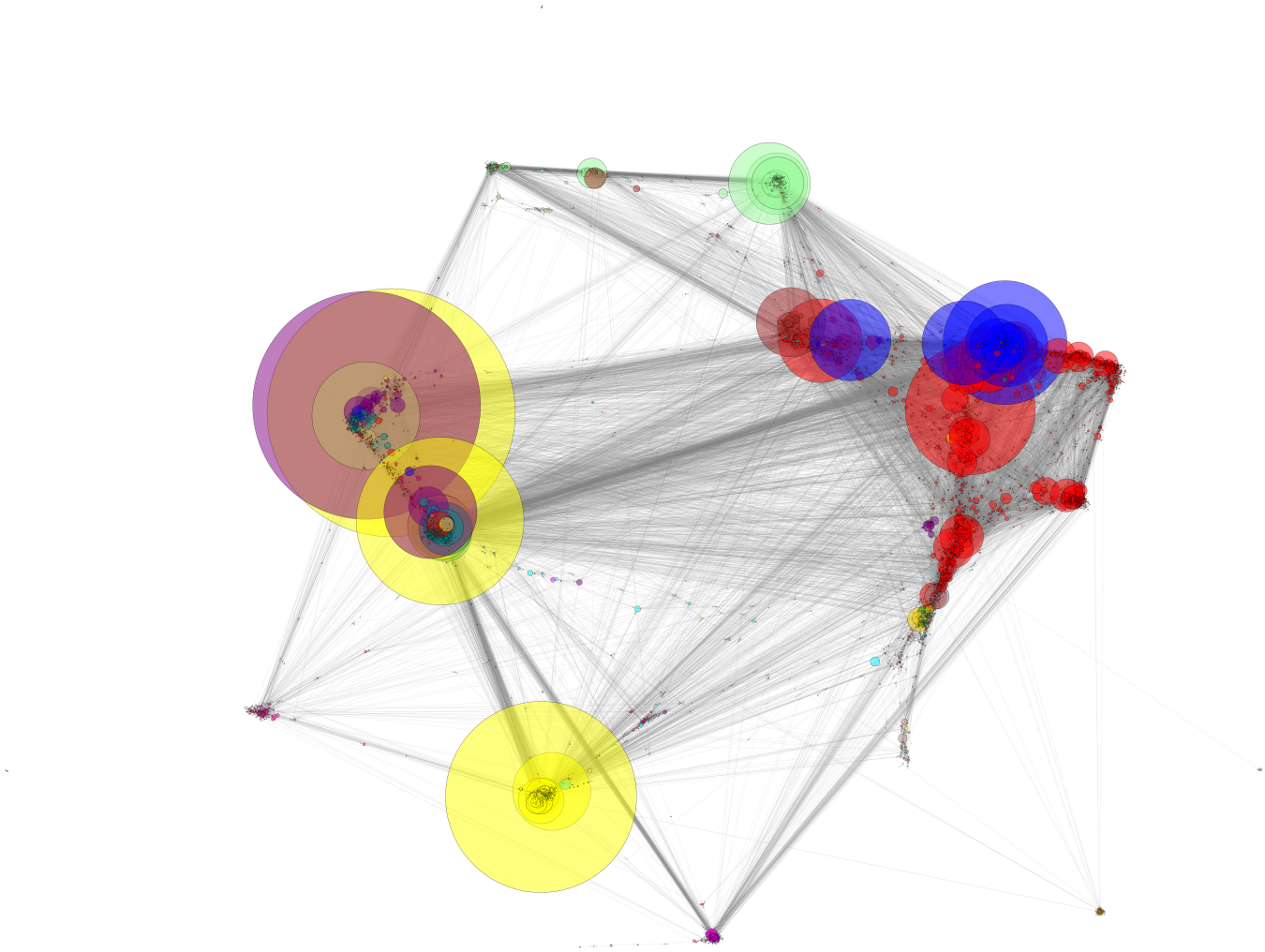

Figure 2: Directed Influence Network after Ranking

3 Music similarity

This part mainly develops measures of music similarity and judges if artists within genre are more similar than artists between genres.

3.1 Similarity Measures

3.1.1 Similarity

According to the articles, there are various methods to compute the similarity between each pair of observations, commonly used methods include Minkowski distance, Mahalannobis distance, several kinds of correlation distances, Hamming distance, Jaccard distance and so on. Different dissimilarity measures are prefered due to the characristics of data and specific problem. Typically, it should be chosen carefully because of its strong influence on the results, especially in some clustering problems.

Correlation-based distance such as Pearson correlation, Eisen cosine correlation, Spearman correlatoin etc. is not suitable for measuring the similarity of tracks, since some fluctuations of a certain characteristic or a group of them are already possible to determine the discrepancy (eg. tempo), even if their features are highly correlated. Besides, Hamming distance and Jaccard distance are apparently inappropriate for this

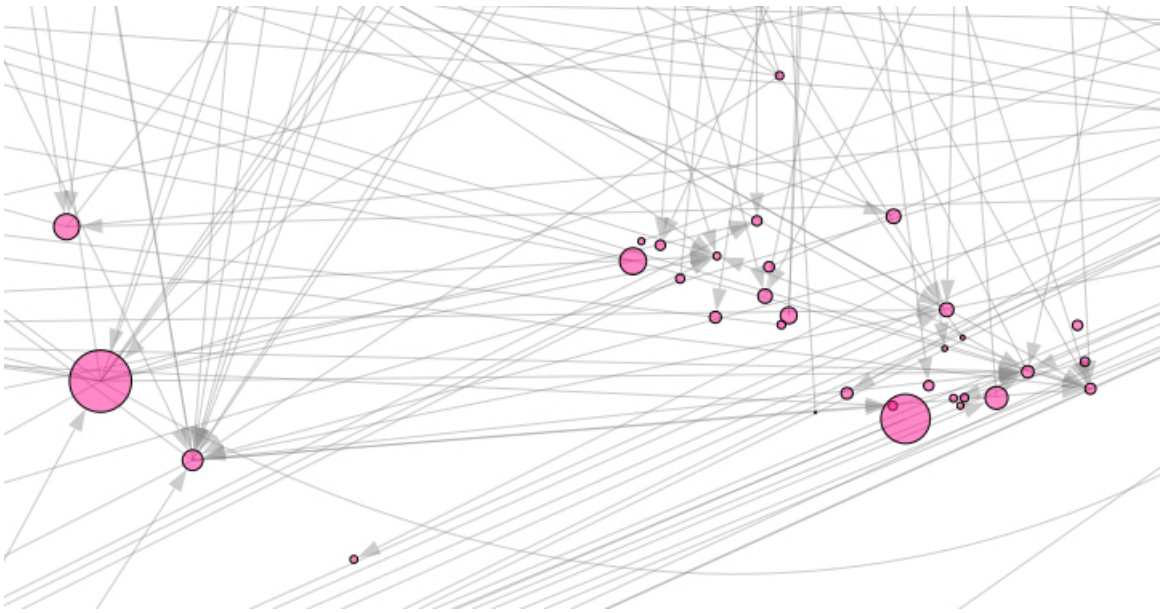

Figure 3: Subnetwork of Latin

problem, so we're considering Minkowski distance and some related variations.

Euclidean distance is the default distance measure for most common clustering software, but it has some obvious flaws while measuring related data or/and data with different scales. However, in the observation and collation of the two summary data sets (with artists and years) of music characteristics as Table 2 shows, features are correlated.

Feature	Type	Range
danceability	float	[0,1]
energy	float	[0,1]
valence	float	[0,1]
tempo	float	[30,207]
loudness	float	[-60,0]
mode	boolean	0/1
key	integer	[0,11]
acousticness	float	[0,1]
instrument	float	[0,1]
liveness	float	[0,1]
speechiness	float	[0,1]
duration_ms	integer	[45707,1640000]
popularity	integer	[0,100]
count	integer	[1,1369]
year	tinyint	[1921,2020]

Table 2: Attributes of Artists

For example, danceability is determined by tempo, rhythm stability, beat strength, and so on. Moreover, scales of features are inconsistent.Based on these, we choose Mahalanobis distance as the similarity index between the music data. Becuase compared

with other measures, Mahalanobis distance corrects the problem that each dimension scale in Euclidean distance is inconsistent and related.

3.1.2 Mahalanobis Distance

We know the original multidimensional sample data $X_{n\times m}$:

$$\begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{pmatrix}$$

Each row represents an artist's data(5854 in total). $x_{ij}$is the j-th feature of the i-th artist. The overall mean of the data is recorded as

$$\mu_x = (\mu_{x_1}, \mu_{x_2}, \cdots, \mu_{x_m})$$

Then its covariance is:

$$\sum\nolimits_X = E(X - \mu_x)^\intercal (X - \mu_x)$$

Firstly, the data points are rootated until the dimensions are linearly independent. Assume the new coordinates are

$$F = (F_1, F_2, \cdots, F_m) = U^\intercal X$$

$$\mu_F = (\mu_{F_1}, \mu_{F_2}, \cdots, \mu_{F_m})$$

$$(F - \mu_F) = U^\intercal (X - \mu_X)$$

After transformation, the dimensions are linearly independent and the variance of each dimension is the eigenvalue.Therefore, it meets the following requirements:

$$\begin{aligned} (F - \mu_F)(F - \mu_F)^\intercal &= \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \cdots & \\ & & & \lambda_m \end{bmatrix} \\ &= U^\intercal (X - \mu_x)(X - \mu_x)^\intercal U \\ &= U^\intercal \sum\nolimits_X U \end{aligned} \tag{3}$$

The Mahalanobis distance is the Euclidean distance after rotation transformation scaling, so the calculation formula of Mahalanobis distance is as follows:

$$\begin{aligned} D &= (\frac{F_1 - \mu_{F_1}}{\sqrt{\lambda_1}})^2 + (\frac{F_2 - \mu_{F_2}}{\sqrt{\lambda_2}})^2 + \cdots + (\frac{F_m - \mu_{F_m}}{\sqrt{\lambda_m}})^2 \\ &= (F - \mu_F)^\intercal (U^\intercal \sum U)^{-1} (F - \mu_F) \\ &= (x - \mu_X)^\intercal \sum\nolimits_X^{-1} (x - \mu_X) \end{aligned} \tag{4}$$

Now, we can use Mahalanobis distance to measure the similarity of different artists.

3.2 Silhouette

Based on the above discussion and access to information, we choose Silhouette to solve the problem. In the area of assessing the quality of clustering, there are generally two criteria: internal evaluation index and external evaluation index, distinguished by whether they use any external information. Our purpose is to judge if artists within genre are more similar than artists between genres. This somewhat matches with the idea of the internal index of clustering that compactness within category, together with separation between categories efficiently indicates the quality of clustering. Thus, we look in a variety of internal indexes to see if any can be used to solve the problem. Fortunately, we find that Silhouette is quite suitable. When it is less than 1 and greater than 0, artists within genre are more similar than artists between genres. Whats more, the larger Silhouette is, the more similar those within the same genre are than those of different genres. Specific derivation is as follows. The artist data has been classified by genre. For artist information point $i \in C_i$ (C_i stands for the i-th genre), it is obvious that C_i is a cluster and an artist's data can be regarded as a data point. Let

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j)$$

be the mean distance between i and all other data points, which belong in the same cluster, where d(i,j) is calculated using the Mahalanobis distance discussed in the previous section. a(i) is interpreted as a measure of how well i is assigned to its cluster. The smaller the value, the better the assignment.

Then we define

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j)$$

to be the smallest mean distance of i to all points in any other cluster, of which i is not a member. The cluster with this smallest mean dissimilarity is said to be the "neighboring cluster" of because it is the next best fit cluster for point .

Next, let

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

,if$|C_i| > 1$ and $s(i) = 0$,if$|C_i| = 1$ be a Sihouette(value) of one data point i. So, we can get the following formula

$$s(i) = \begin{cases} 1 - a(i)/b(i), if & a(i) < b(i) \\ 0, if & a(i) = b(i) \\ b(i)/a(i) - 1, if & a(i) > b(i) \end{cases}$$

It is clear that -1≤s(i)≤1. Also, 0 is added to prevent the number of clusters from increasing significantly. A small a(i) means it is well matched and a large b(i) implies that i is badly matched to its neighboring cluster. Thus when the data is appropriately clustered, s(i) close to one. If s(i) is close to negative one, it would be more appropriate if it was clustered in its neighboring cluster. An s(i) near zero means that the datum is

on the border of two natural clusters. Based on the above discussion, when s(i) is in the range of 0 to 1, the conclusion is proved.

Then look for the maximum value of the mean s(i) in the same cluster. By programming, we can verify that for any genre, the maximum value of the mean s(i) is greater than 0 and less than 1. This proves that artists within genre are more similar than artists between genres.

References

[1] Liu, J. Q. , Li, X. R. , & Dong, J. C. (2020). A survey on network node ranking algorithms: representative methods, extensions, and applications. Science China Technological Sciences, 1-11.

[2] Brin, S. , & Page, L. (2012). Reprint of: the anatomy of a large-scale hypertextual web search engine. Computer Networks, 56(18), 3825-3833.

[3] Page, L. (1998). The pagerank citation ranking: bringing order to the web, online manuscript. http://www-db.stanford.edu/~backrub/pageranksub.ps.

[4] Janice, W. (2011, April 26). Visualising music: the problems with genre classification. *Masters of Media*. Retrieved February 8, 2021, from http://mastersofmedia.hum.uva.nl/blog/2011/04/26/visualising-music-the-problems-with-genre-classification/♯

[5] Yu, Z. , Yudong, L. , & Zhao, J. (2009). Vector similarity measurement method. Technical Acoustics (04), 532-536.

[6] Peter, J., & Rousseeuw. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics.

[7] De Amorim, R. C. , & Hennig, C. (2015). Recovering the number of clusters in data sets with noise features using feature rescaling factors. Information Sciences, 324, 126-145.

[8] Kaufman, L. , & Rousseeuw, P. J. (2005). Finding groups in data: an introduction to cluster analysis. Wiley.

Appendices

Appendix A Code for Part 2

```
######################
## Task 1
######################
import time
import pandas as pd
import networkx as nx
```

```
import igraph as ig
from networkx import NetworkXError

color_dict = {'Pop/Rock': 'rgba(255,0,0,0.5)',
            'Electronic': 'rgba(128,128,128,0.5)',
            'Reggae': 'rgba(255,0,255,0.5)',
            'Jazz': 'rgba(255,255,0,0.5)',
            'Country': 'rgba(152,251,152,0.5)',
            'Comedy/Spoken': 'rgba(255,165,0,0.5)',
            'R&B;': 'rgba(0,139,139,0.5)',
            'Latin': 'rgba(255,20,147,0.5)',
            'Vocal': 'rgba(128,0,128,0.5)',
            'Folk': 'rgba(165,42,42,0.5)',
            'Easy Listening': 'rgba(127,255,212,0.5)',
            'International': 'rgba(0,255,255,0.5)',
            'Avant-Garde': 'rgba(0,128,0,0.5)',
            'Blues': 'rgba(0,0,255,0.5)',
            'Classical': 'rgba(255,215,0,0.5)',
            'Stage & Screen': 'rgba(188,143,143,0.5)',
            'New Age': 'rgba(139,0,0,0.5)',
            'Religious': 'rgba(189,183,107,0.5)',
            "Children's": 'rgba(240,255,255,0.5)',
            'Unknown': 'rgba(0,0,0,0.5)'
            }

nwx = nx.DiGraph()
nwx.name = {}
nwx.main_gengre = {}
nwx.active_start = {}

df = pd.read_csv("influence_data.csv")
print(df.head())

for rowindex in df.index.tolist():
    id = df.iat[rowindex, 0]
    nwx.add_node(id)
    nwx.name[id] = df.iat[rowindex, 1]
    nwx.main_gengre[id] = df.iat[rowindex, 2]
    nwx.active_start[id] = df.iat[rowindex, 3]
    id = df.iat[rowindex, 4]
    nwx.add_node(id)
    nwx.name[id] = df.iat[rowindex, 5]
    nwx.main_gengre[id] = df.iat[rowindex, 6]
    nwx.active_start[id] = df.iat[rowindex, 7]
    nwx.add_edge(df.iat[rowindex, 0], df.iat[rowindex, 4])


g = ig.Graph.from_networkx(nwx)


visual_style = {}
visual_style["vertex_size"] = 10
visual_style["vertex_color"] = [color_dict[nwx.main_gengre[v]] for v in nwx]
visual_style["arrow_width"] = 1
visual_style["arrow_size"] = 1
#visual_style["vertex_label"] = [nwx.name[v] for v in nwx]
visual_style["edge_width"] = 1
visual_style["edge_color"] = 'rgba(128,128,128,0.25)'
```

```
visual_style["layout"] = g.layout("drl")
visual_style["bbox"] = (900, 600)
visual_style["margin"] = 20
ig.plot(g, **visual_style)


def pagerank(G, alpha=0.85, personalization=None,
             max_iter=100, tol=1.0e-6, nstart=None, weight='weight',
             dangling=None):
    """Return the PageRank of the nodes in the graph.

    PageRank computes a ranking of the nodes in the graph G based on
    the structure of the incoming links. It was originally designed as
    an algorithm to rank web pages.

    Parameters
    ----------
    G : graph
      A NetworkX graph.  Undirected graphs will be converted to a directed
      graph with two directed edges for each undirected edge.

    alpha : float, optional
      Damping parameter for PageRank, default=0.85.

    personalization: dict, optional
      The "personalization vector" consisting of a dictionary with a
      key for every graph node and nonzero personalization value for each node.
      By default, a uniform distribution is used.

    max_iter : integer, optional
      Maximum number of iterations in power method eigenvalue solver.

    tol : float, optional
      Error tolerance used to check convergence in power method solver.

    nstart : dictionary, optional
      Starting value of PageRank iteration for each node.

    weight : key, optional
      Edge data key to use as weight.  If None weights are set to 1.

    dangling: dict, optional
      The outedges to be assigned to any "dangling" nodes, i.e., nodes without
      any outedges. The dict key is the node the outedge points to and the dict
      value is the weight of that outedge. By default, dangling nodes are given
      outedges according to the personalization vector (uniform if not
      specified). This must be selected to result in an irreducible transition
      matrix (see notes under google_matrix). It may be common to have the
      dangling dict to be the same as the personalization dict.

    Returns
    -------
    pagerank : dictionary
       Dictionary of nodes with PageRank as value

    Notes
    -----
    The eigenvector calculation is done by the power iteration method
```

```
    and has no guarantee of convergence.  The iteration will stop
    after max_iter iterations or an error tolerance of
    number_of_nodes(G)*tol has been reached.

    The PageRank algorithm was designed for directed graphs but this
    algorithm does not check if the input graph is directed and will
    execute on undirected graphs by converting each edge in the
    directed graph to two edges.


    """
    if len(G) == 0:
        return {}

    if not G.is_directed():
        D = G.to_directed()
    else:
        D = G

        # Create a copy in (right) stochastic form
    W = nx.stochastic_graph(D, weight=weight)
    N = W.number_of_nodes()

    # Choose fixed starting vector if not given
    if nstart is None:
        x = dict.fromkeys(W, 1.0 / N)
    else:
        # Normalized nstart vector
        s = float(sum(nstart.values()))
        x = dict((k, v / s) for k, v in nstart.items())

    if personalization is None:

        # Assign uniform personalization vector if not given
        p = dict.fromkeys(W, 1.0 / N)
    else:
        missing = set(G) - set(personalization)
        if missing:
            raise NetworkXError('Personalization dictionary '
                                'must have a value for every node. '
                                'Missing nodes %s' % missing)
        s = float(sum(personalization.values()))
        p = dict((k, v / s) for k, v in personalization.items())

    if dangling is None:

        # Use personalization vector if dangling vector not specified
        dangling_weights = p
    else:
        missing = set(G) - set(dangling)
        if missing:
            raise NetworkXError('Dangling node dictionary '
                                'must have a value for every node. '
                                'Missing nodes %s' % missing)
        s = float(sum(dangling.values()))
        dangling_weights = dict((k, v / s) for k, v in dangling.items())
    dangling_nodes = [n for n in W if W.out_degree(n, weight=weight) == 0.0]
```

```
    # power iteration: make up to max_iter iterations
    for _ in range(max_iter):
        xlast = x
        x = dict.fromkeys(xlast.keys(), 0)
        danglesum = alpha * sum(xlast[n] for n in dangling_nodes)
        for n in x:

            # this matrix multiply looks odd because it is
            # doing a left multiply x^T=xlast^T*W
            for nbr in W[n]:
                x[nbr] += alpha * xlast[n] * W[n][nbr][weight]
            x[n] += danglesum * dangling_weights[n] + (1.0 - alpha) * p[n]

            # check convergence, l1 norm
        err = sum([abs(x[n] - xlast[n]) for n in x])
        if err < N * tol:
            return x
    raise NetworkXError('pagerank: power iteration failed to converge '
                        'in %d iterations.' % max_iter)

start_year = 1921 - 1
years = list(nwx.active_start.values())
tmp = [0] * (2020-1921+1+1)
for i in range(len(years)):
    tmp[years[i]-start_year] = tmp[years[i]-start_year] + 1
cnt = 0
for i in range(len(tmp)):
    cnt = cnt + tmp[i]
    tmp[i] = cnt
ranks = []
for i in range(len(nwx.nodes)):
    ranks.append(tmp[years[i] - start_year])
per = dict(zip(nwx.nodes, ranks))

rnwx = nwx.reverse()
pg = pagerank(rnwx, 0.85, personalization=per)
sorted_pg = sorted(pg.items(), key=lambda d: d[1], reverse=True)
print(sorted_pg)
vs = [i * 70000 for i in list(pg.values())]
visual_style["vertex_size"] = vs
visual_style["bbox"] = (6400, 4800)

start = time.clock()
ig.plot(g, **visual_style)
elapsed = (time.clock() - start)
print("Time used:",elapsed)

gengres = list(nwx.main_gengre.values())
names = list(nwx.name.values())

# subv = [i for i,x in enumerate(gengres) if x == 'Jazz']
# jazz_graph = g.subgraph(subv)
# visual_style["vertex_label"] = [names[i] for i in subv]
# vs = [i * 1e6 for i in list(pg.values())]
# visual_style["vertex_size"] = [vs[i] for i in subv]
# visual_style["vertex_color"] = [color_dict[gengres[i]] for i in subv]
# ig.plot(jazz_graph, **visual_style)
```

```
visual_style["edge_color"] = 'rgba(128,128,128,0.4)'

subv = [i for i,x in enumerate(gengres) if x == 'Country']
country_graph = g.subgraph(subv)
#visual_style["vertex_label"] = [names[i] for i in subv]
vs = [i * 1e5 for i in list(pg.values())]
visual_style["vertex_size"] = [vs[i] for i in subv]
visual_style["vertex_color"] = [color_dict[gengres[i]] for i in subv]
ig.plot(country_graph, **visual_style)

subv = [i for i,x in enumerate(gengres) if x == 'Pop/Rock']
pr_graph = g.subgraph(subv)
#visual_style["vertex_label"] = [names[i] for i in subv]
vs = [i * 5 * 1e4 for i in list(pg.values())]
visual_style["vertex_size"] = [vs[i] for i in subv]
visual_style["vertex_color"] = [color_dict[gengres[i]] for i in subv]
ig.plot(pr_graph, **visual_style)

subv = [i for i,x in enumerate(gengres) if x == 'Latin']
latin_graph = g.subgraph(subv)
vs = [i * 1e5 for i in list(pg.values())]
visual_style["vertex_size"] = [vs[i] for i in subv]
visual_style["vertex_color"] = [color_dict[gengres[i]] for i in subv]
ig.plot(latin_graph, **visual_style)
```