In []: import pandas as pd import networkx as nx import matplotlib.pyplot as plt import matplotlib.colors as mcolors #import plotly.graph_objects as go from numpy import linalg as LA from collections import defaultdict import numpy as np import random from community import community_louvain from graphrole import RecursiveFeatureExtractor, RoleExtractor import multiprocessing as mp import itertools from numpy import linalg as LA from os import cpu_count import time from modelos import configuration_model from metricas import distribucion_grados from motifs_calculos import motif_grafo_eleatorios from motifs.graficos import graficar_significant_profile from motifs_calculos import significance_profile In [8]: world = pd.read_csv("World.csv") world = world.drop(['ConexionAeropuertos'], axis=1) world Out[8]: Destino Origen 0 Papua New Guinea Australia 1 Papua New Guinea Philippines 2 Papua New Guinea Indonesia 3 Papua New Guinea Solomon Islands 4 Papua New Guinea Hong Kong 2847 Lithuania Georgia 2848 Armenia Georgia 2849 Eritrea Yemen 2850 Yemen Djibouti 2851 Solomon Islands Nauru 2852 rows × 2 columns Ejercicio 1. a. Obtener una visualización de las comunidades presentes en dicha red (indicando el algoritmo utilizado). b. Considerando lo que respondiste en el parcialito 1 (ejercicio 2): i. Si mencionaste que había homofilia, ¿corresponde por el mismo tipo que mencionaste anteriormente? ¿por qué? ii. Si mencionaste que no había homofilia (o bien no realizaste el ejercicio), ¿qué tipo de homofilia se puede ver presente? c. Obtener los nodos correspondientes a una de las subredes (con al menos 20% de los nodos), y realizar una visualización de las sub-comunidades presentes. In [9]: ## Genero el grafo que va a representar la red. ## grafo = nx.from_pandas_edgelist(world, 'Origen', 'Destino') In []: # Ejercicio 1.a algoritmo basado al encontrado en: https://stackoverflow.com/questions/43541376/how-to-draw-communities-with-networkx def community_layout(g, partition): pos_communities = _position_communities(g, partition, scale=3.) pos_nodes = _position_nodes(g, partition, scale=1.) pos = dict() for node in g.nodes(): pos[node] = pos_communities[node] + pos_nodes[node] return pos def _position_communities(g, partition, **kwargs): between_community_edges = _find_between_community_edges(g, partition) communities = set(partition.values()) hypergraph = nx.DiGraph()hypergraph.add_nodes_from(communities) for (ci, cj), edges in between_community_edges.items(): hypergraph.add_edge(ci, cj, weight=len(edges)) pos_communities = nx.spring_layout(hypergraph, **kwargs) pos = dict() for node, community in partition.items(): pos[node] = pos_communities[community] return pos def _find_between_community_edges(g, partition): edges = dict() for (ni, nj) in g.edges(): ci = partition[ni] cj = partition[nj] **if** ci != cj: try: edges[(ci, cj)] += [(ni, nj)]except KeyError: edges[(ci, cj)] = [(ni, nj)]return edges def _position_nodes(g, partition, **kwargs): communities = dict() for node, community in partition.items(): try: communities[community] += [node] except KeyError: communities[community] = [node] pos = dict() for ci, nodes in communities.items(): subgraph = g.subgraph(nodes) pos_subgraph = nx.spring_layout(subgraph, **kwargs) pos.update(pos_subgraph) return pos def graficar_comunidades(g): partition = community_louvain.best_partition(g) pos = community_layout(g, partition) nx.draw_networkx(g, pos, node_color=list(partition.values()), with_labels=True); plt.rcParams["figure.figsize"] = (20,8) plt.show() return graficar_comunidades(grafo) Western Sahar Ejercicio 1 b. El resultado coincide con lo obtenido en el primer parcialito. Las comunidades en este grafo pertenecen a continentes o "lugares geograficos". Por ejemplo america y caribe se lo podria ver como una comunidad. Esto tiene mucho sentido ya que los vuelos suelen ser mas frecuentes entre paises limitrofes o de zonas cercanas. El algoritmo utilizado es Louvain. In []: # *Ejercicio 1.c* comunidades = community_louvain.best_partition(grafo) grupo_asia_oceania = dict() for key, value in comunidades.items(): if value == 0: grupo_asia_oceania[key] = value grafo_asia_oceania = grafo In []: for node in list(grafo_asia_oceania.nodes): if grupo_asia_oceania.get(node) == None: grafo_asia_oceania.remove_node(node) graficar_comunidades(grafo_asia_oceania) Los subgrupos en esta red tambien estan muy marcados. Este subgrupo se lo podria denominar como un grupo compuesto por paises de Asia y Oceania. Donde los subgrupos serian los lugares o zonas geograficas. Por ejemplo todos los paises oceanicos estan en un grupo, los asiaticos-orientales por otro y finalmente el ultimo los asiaticos arabes y ex union sovietica. Ejercicio 2 a. Calcular los motifs de hasta 5 nodos de la subred definida en el punto 1.c. In []: from motifs_calculos import calcular_motifs motifs = calcular_motifs(grafo_asia_oceania, 5) print("motifs: ", motifs) motifs: [7190 2099 39996 34368 2320 56367 21333 5458 128705 356878 106317 259268 186957 312914 3839 46909 397100 61354 227101 2900 39091 73231 231422 178720 16627 180573 15283 57637 9945] b. Calcular el promedio y desvío estandar de los motifs de una red de baseline. Calcular el significant profile de la red, y hacer un gráfico. In []: dist = distribucion_grados(grafo_asia_oceania) promedios, stds = motif_grafo_eleatorios(lambda: configuration_model(dist), 5, iters=10) In []: print("promedios: ", promedios) promedios: [7.275700e+03 6.766000e+02 5.939560e+04 3.927540e+04 1.050200e+04 2.889760e+04 6.341700e+03 4.110000e+02 3.167841e+05 6.024827e+05 1.250220e+05 2.393289e+05 1.226296e+05 1.909310e+05 4.884290e+04 3.151011e+05 1.963821e+05 1.502440e+04 1.224080e+05 3.592290e+04 1.426141e+05 1.686360e+04 3.126000e+04 7.165200e+04 4.909100e+04 2.050880e+04 1.090070e+04 3.197000e+03 9.510000e+01] In []: print("desvios estándar: ", stds) desvios estándar: [3.45796776e+02 6.19373877e+01 2.30226429e+03 3.33438091e+03 6.38837068e+02 2.91628727e+03 9.68815777e+02 7.99512351e+01 9.37392883e+03 3.55447131e+04 1.75117146e+04 2.41356254e+04 8.17981880e+03 2.52323077e+04 2.90324362e+03 1.92228773e+04 3.28034126e+04 2.00929835e+03 1.49813249e+04 3.89121437e+03 1.28937099e+04 4.28043475e+03 5.83141719e+03 1.37826754e+04 6.01486036e+03 5.04761858e+03 2.59963201e+03 9.50796298e+02 3.52035510e+01] In []: SP = significance_profile(motifs, promedios, stds) print("SP:", SP) SP: [-8.22394144e-04 7.62049583e-02 -2.79613923e-02 -4.88379583e-03 -4.25001115e-02 3.12564827e-02 5.13474673e-02 2.09471083e-01 -6.65794529e-02 -2.29288900e-02 -3.54446262e-03 2.74137445e-03 2.60959820e-02 1.60421918e-02 -5.14384536e-02 -4.62965840e-02 2.03043199e-02 7.65130105e-02 2.31893607e-02 -2.81612108e-02 -2.66428527e-02 4.36979842e-02 1.13901338e-01 2.57778988e-02-1.79100909e-02 1.05227433e-01 5.59385808e-03 1.89999107e-019.28441180e-01] In []: graficar_significant_profile(SP, 'Asia-Oceania') Significant profile per motif - Asia-Oceania 0.8 Significance Profile 0.6 0.4 0.2 0.0 1 2 3 4 5 6 7 8 91011121314151617181920212223242526272829 c. Intentar dar con una explicación del resultado obtenido en el punto anterior [+1 punto]. Ejercicio 3 Detectar los roles en dicha red utilizando el algoritmo RolX, explicando el resultado obtenido. In []: ''' algoritmo basado al encontrado en: https://drive.google.com/file/d/11heexfkwbJdl8TPJHnbEic4RdVzFMeR6/view?usp=sharing def extract_roles_and_plot(G, title='', save = False , file_name='', big=False): feature_extractor = RecursiveFeatureExtractor(G) features = feature_extractor.extract_features() role_extractor = RoleExtractor(n_roles=None) role_extractor.extract_role_factors(features) labels = {node: node for node in G.nodes()} available_colors = \ {'role_0':'#E9D758', 'role_1':'#297373', 'role_2':'#ff8552','role_3':'#888888', 'role_4' :'#00aa00','role_5':'#aaaa00','role_6':'#aa0000' , 'role_7':'#0000aa'} colors = [available_colors[role_extractor.roles[node]] for node in G.nodes()] pos = nx.kamada_kawai_layout(G) if big: plt.figure(figsize=(25,25)) else: plt.figure(figsize=(20,20)) plt.title(title) nx.draw_networkx_nodes(G, pos, nodelist=G.nodes(), node_color=colors, alpha=0.7, node_siz e=200, linewidths=2) nx.draw_networkx_edges(G, pos, width=0.3, alpha=0.05) nx.draw_networkx_labels(G, pos, labels=labels) if save: plt.savefig(file_name, format = 'svg', dpi=300) plt.figure(figsize=(10,10)) plt.title(title) nx.draw_networkx_nodes(G, pos, nodelist=G.nodes(), node_color=colors, alpha=0.7, node_siz e=200, linewidths=2) nx.draw_networkx_edges(G, pos, width=0.3, alpha=0.5) nx.draw_networkx_labels(G, pos, labels=labels) plt.savefig(file_name, format = 'svg', dpi=300) In []: grafo = nx.from_pandas_edgelist(world, 'Origen', 'Destino') extract_roles_and_plot(grafo, title='Roles', save=False, file_name='img/roles_ep4.svg') Cocos (Keeling) Islands Como el grafico no es del todo entendible (hay una gran cantidad de nodos, quizas graficando un subgrupo se hace mas entendible el grafico) voy a ver que paises pertenecen a cada rol para hacer el analisis. In [13]: | feature_extractor = RecursiveFeatureExtractor(grafo) features = feature_extractor.extract_features() role_extractor = RoleExtractor(n_roles=None) role_extractor.extract_role_factors(features) In [14]: paises_por_roles = [[], [], [], [], [], [], []] for i in range(0,5): for key, value in role_extractor.roles.items(): if i == int(value[len(value)-1]): paises_por_roles[i].append(key) for i in range(0,5): print('Los paises pertenecientes al rol', i, 'son:') for value in paises_por_roles[i]: print(value, end=', ') print('\n') Los paises pertenecientes al rol 0 son: Austria, Cote d'Ivoire, Cyprus, Czech Republic, Denmark, Egypt, Greece, Hungary, India, Ital y, Malaysia, Malta, Norway, Poland, Russia, Sweden, Switzerland, Thailand, Ukraine, United S tates,

Los paises pertenecientes al rol 1 son:

Los paises pertenecientes al rol 2 son:

Los paises pertenecientes al rol 3 son:

Los paises pertenecientes al rol 4 son:

ngdom,

In []:

Micronesia, Solomon Islands, Wallis and Futuna,

Afghanistan, American Samoa, Anguilla, Antigua and Barbuda, Armenia, Aruba, Bahamas, Barbado s, Belize, Benin, Bermuda, Bhutan, Bolivia, Botswana, British Virgin Islands, Brunei, Burkin a Faso, Burma, Burundi, Cambodia, Cameroon, Cayman Islands, Central African Republic, Chad, Christmas Island, Cocos (Keeling) Islands, Comoros, Congo (Brazzaville), Congo (Kinshasa), C ook Islands, Djibouti, Dominica, East Timor, Ecuador, El Salvador, Equatorial Guinea, Eritre a, Falkland Islands, Faroe Islands, Fiji, French Guiana, French Polynesia, Gabon, Gambia, Gi braltar, Greenland, Grenada, Guadeloupe, Guam, Guatemala, Guernsey, Guinea, Guinea-Bissau, G uyana, Haiti, Honduras, Isle of Man, Jamaica, Jersey, Kiribati, Kyrgyzstan, Laos, Lesotho, L iberia, Libya, Macau, Madagascar, Malawi, Mali, Marshall Islands, Martinique, Mauritania, Ma yotte, Mongolia, Montserrat, Mozambique, Myanmar, Namibia, Nauru, Nepal, Netherlands Antille s, New Caledonia, New Zealand, Nicaragua, Niger, Niue, Norfolk Island, North Korea, Northern Mariana Islands, Palau, Papua New Guinea, Paraguay, Peru, Puerto Rico, Reunion, Rwanda, Sain t Helena, Saint Kitts and Nevis, Saint Lucia, Saint Pierre and Miquelon, Saint Vincent and t he Grenadines, Samoa, Sao Tome and Principe, Sierra Leone, Somalia, South Sudan, Sudan, Suri name, Swaziland, Syria, Tajikistan, Togo, Tonga, Trinidad and Tobago, Turkmenistan, Turks an d Caicos Islands, Tuvalu, Uruguay, Vanuatu, Virgin Islands, Western Sahara, Yemen, Zambia, Z

Albania, Algeria, Angola, Argentina, Australia, Azerbaijan, Bahrain, Bangladesh, Belarus, Bo snia and Herzegovina, Brazil, Bulgaria, Canada, Cape Verde, Chile, China, Colombia, Costa Ri ca, Croatia, Cuba, Dominican Republic, Estonia, Ethiopia, Finland, Georgia, Ghana, Hong Kon g, Iceland, Indonesia, Iran, Iraq, Ireland, Israel, Japan, Jordan, Kazakhstan, Kenya, Kuwai t, Latvia, Lebanon, Lithuania, Luxembourg, Macedonia, Maldives, Mauritius, Mexico, Moldova, Montenegro, Morocco, Nigeria, Oman, Pakistan, Panama, Philippines, Portugal, Romania, Saudi Arabia, Senegal, Serbia, Seychelles, Singapore, Slovakia, Slovenia, South Africa, South Kore

Belgium, France, Germany, Netherlands, Qatar, Spain, Turkey, United Arab Emirates, United Ki

Combinando la informacion recien impresa y el grafico podemos determinar que los paises pertenecientes a los roles 1 y 3 son los nodos perifericos. Lo cual tiene sentido ya que son los paises con menor conexion de vuelos. Por otra parte, los nodos del rol 4 (color verde claro) son los nodos centrales. Estos paises son de los mas desarrollados, con mejor ubicacion

a, Sri Lanka, Taiwan, Tanzania, Tunisia, Uganda, Uzbekistan, Venezuela, Vietnam,

y lugares estrategicos claves para la conexion de muchos vuelos a todos los paises del mundo.