arizona

May 6, 2021

https://github.com/Toblerity/Fiona/issues/944

[1]:
```
import fiona
```

[2]:
```
import matplotlib.pyplot as plt
from gerrychain import (GeographicPartition, Partition, Graph, MarkovChain,
                        proposals, updaters, constraints, accept, Election)
from gerrychain.proposals import recom
from functools import partial
import pandas
```

[3]:
```
import maup
import numpy as np
import geopandas
import matplotlib.pyplot as plt
from gerrychain import (GeographicPartition, Partition, Graph, MarkovChain,
                        proposals, updaters, constraints, accept, Election)
from gerrychain.updaters import Tally, cut_edges, exterior_boundaries,␣
 ↪exterior_boundaries_as_a_set
from networkx import is_connected, connected_components
```

[4]:
```
graph = Graph.from_file("./az_precincts.zip", ignore_errors=True)
precincts = geopandas.read_file("./az_precincts.zip")
```

```
C:\Users\darre\anaconda3\envs\gerrychain\lib\site-
packages\gerrychain\graph\adjacency.py:66: UserWarning: Found overlaps among the
given polygons. Indices of overlaps: {(153, 617), (61, 235), (224, 241), (991,
1081), (101, 1482), (354, 637), (554, 1089), (715, 768), (1034, 1388), (27,
132), (860, 1281), (533, 767), (119, 215), (36, 89), (88, 127), (82, 175), (106,
532), (410, 516), (578, 637), (118, 524), (615, 1330), (39, 1292), (143, 318),
(140, 420), (331, 1400), (76, 127), (372, 672), (421, 1431), (19, 377), (66,
974), (134, 343), (329, 710), (380, 532), (799, 1400), (307, 389), (608, 1430),
(102, 377), (54, 1112), (48, 177), (66, 848), (39, 1379), (555, 1024), (84,
246), (1201, 1434), (291, 447), (1208, 1412), (1377, 1388), (503, 723), (421,
651), (1053, 1383), (300, 352), (433, 1400), (1252, 1363), (185, 285), (676,
1316), (39, 1372), (340, 406), (427, 1487), (991, 1181), (42, 212), (488, 1482),
(29, 669), (81, 978), (444, 1039), (451, 1481), (647, 1246), (727, 1126), (351,
439), (253, 532), (14, 21), (43, 57), (752, 1457), (831, 1174), (1134, 1373),
(14, 158), (76, 618), (66, 1218), (518, 860), (848, 1391), (28, 1481), (155,
```

420), (162, 1481), (421, 500), (421, 1046), (1144, 1252), (39, 318), (179, 219), (27, 243), (179, 338), (0, 345), (5, 66), (181, 420), (1156, 1244), (80, 197), (991, 1039), (416, 1333), (666, 742), (683, 1298), (842, 1233), (483, 1371), (187, 326), (771, 1310), (831, 904), (102, 1441), (735, 1363), (251, 347), (604, 1389), (669, 1271), (48, 158), (1272, 1430), (1, 1129), (1283, 1430), (139, 395), (57, 115), (134, 326), (309, 318), (11, 246), (936, 1380), (45, 516), (135, 153), (733, 1310), (119, 1487), (122, 610), (378, 1479), (76, 622), (819, 1478), (571, 711), (502, 1267), (14, 1193), (331, 711), (555, 1389), (672, 742), (831, 1461), (120, 674), (593, 1034), (637, 674), (19, 115), (43, 1485), (69, 610), (275, 1481), (484, 581), (13, 724), (651, 1350), (808, 1315), (995, 1013), (111, 1485), (82, 674), (353, 1009), (1357, 1393), (1034, 1377), (22, 244), (841, 1339), (848, 1116), (8, 593), (1081, 1357), (14, 267), (0, 488), (377, 1112), (602, 712), (808, 899), (46, 76), (428, 1322), (14, 233), (614, 942), (703, 1270), (345, 488), (150, 1472), (320, 434), (66, 1082), (334, 1400), (293, 1481), (330, 484), (26, 243), (623, 1127), (863, 1383), (475, 1485), (672, 1285), (717, 899), (43, 1482), (39, 1103), (94, 899), (173, 408), (516, 610), (1390, 1412), (410, 610), (11, 593), (780, 1232), (1134, 1315), (27, 201), (66, 1212), (31, 43), (426, 472), (448, 502), (1178, 1379), (1200, 1379), (421, 1150), (10, 267), (80, 1485), (552, 1034), (670, 1198), (59, 416), (983, 1429), (659, 1326), (1046, 1049), (677, 685), (807, 1331), (386, 502), (1012, 1193), (1174, 1390), (365, 836), (1486, 1488), (738, 1326), (1081, 1201), (101, 488), (779, 979), (647, 1226), (293, 420), (319, 448), (42, 1479), (15, 85), (114, 326), (59, 119), (360, 1123), (39, 936), (90, 780), (1169, 1286), (43, 787), (82, 120), (987, 1129), (1201, 1357), (127, 408), (254, 488), (536, 704), (17, 542), (99, 1487), (53, 318), (120, 1400), (1328, 1434), (637, 1400), (221, 448), (496, 1043), (1377, 1457), (122, 179), (325, 496), (770, 1264), (320, 1482), (521, 1472), (556, 664), (14, 771), (120, 362), (39, 1212), (949, 1403), (94, 1246), (9, 412), (19, 50), (74, 532), (502, 1467), (514, 932), (921, 1051), (208, 212), (794, 1096), (31, 1482), (82, 472), (122, 419), (181, 386), (1051, 1127), (39, 421), (488, 1059), (421, 969), (115, 377), (940, 1305), (969, 1350), (9, 533), (88, 408), (651, 1031), (74, 253), (779, 969), (43, 1467), (472, 656), (983, 1241), (185, 724), (800, 1156), (426, 567), (942, 1078), (1055, 1165), (332, 532), (115, 116), (848, 1089), (1146, 1411), (22, 345), (59, 212), (39, 1200), (491, 1190), (420, 1267), (254, 1482), (433, 581), (1051, 1378), (83, 1481), (52, 224), (674, 1400), (1097, 1146), (45, 185), (630, 1223), (643, 1461), (219, 340), (217, 395), (25, 362), (514, 1066), (770, 1144), (18, 167), (644, 1285), (185, 472), (1133, 1214), (1046, 1274), (14, 590), (45, 724), (992, 1312), (1089, 1263), (329, 730), (90, 1232), (629, 723), (1143, 1362), (235, 1232), (1173, 1363), (39, 1049), (85, 488), (770, 1375), (626, 711), (637, 711), (217, 244), (98, 216), (320, 1321), (90, 212), (1057, 1471), (209, 1481), (349, 1485), (128, 205), (618, 1271), (39, 1170), (43, 118), (1, 2), (285, 426), (759, 840), (8, 17), (421, 979), (14, 48), (153, 243), (949, 979), (666, 767), (197, 1481), (251, 406), (530, 733), (1224, 1449), (1326, 1478), (56, 1112), (285, 419), (167, 183), (201, 1317), (1039, 1403), (338, 419), (45, 285), (2, 76), (734, 1322), (97, 841), (731, 1015), (39, 500), (434, 502), (614, 1244), (39, 1284), (936, 1414), (968, 1327), (334, 433), (420, 466), (571, 626), (709, 1243), (717, 808), (150, 352), (11, 17), (134, 1000), (899, 1373), (518, 736), (815, 1123), (224, 408), (56, 415), (1155, 1198), (836, 1045), (59, 359), (45,

```
305), (189, 1481), (362, 1400), (39, 1414), (602, 753), (43, 124), (672, 1187),
(9, 724), (752, 1216), (340, 347), (489, 1379), (243, 1418), (35, 1270), (636,
1123), (807, 874), (212, 1479), (569, 694), (518, 1412), (1174, 1290), (22,
305), (0, 1059), (14, 63), (643, 1156), (874, 1040), (421, 933), (227, 1467),
(2, 1129), (522, 637), (561, 1373), (2, 46), (779, 1434), (389, 730), (554,
1334), (1428, 1471), (963, 1243), (467, 1305), (1073, 1133), (581, 799)}
  warnings.warn(
```

[5]:
```
components = list(connected_components(graph))
biggest_component_size = max(len(c) for c in components)
problem_components = [c for c in components if len(c) != biggest_component_size]
for component in problem_components:
    for node in component:
        graph.remove_node(node)
is_connected(graph)
```

[5]:
```
True
```

[6]:
```
election = Election("PRES16", {"Dem": "PRES16D", "Rep": "PRES16R"})

initial_partition = GeographicPartition(
    graph,
    assignment="CD",
    updaters={
        "cut_edges": cut_edges,
        "population": Tally("TOTPOP", alias="population"),
        "PRES16": election
    }
)
```

[7]:
```
from gerrychain.constraints.contiguity import contiguous_components, contiguous
from gerrychain import Partition
bad_nodeview = contiguous_components(initial_partition).get('01')[1].nodes()
bad_nodeview
```

[7]:
```
NodeView((208,))
```

[9]:
```
graph
```

[9]:
```
<Graph [1489 nodes, 4096 edges]>
```

[10]:
```
contiguous_components(initial_partition)
```

[10]:
```
{'01': [<Graph [278 nodes, 723 edges]>, <Graph [1 nodes, 0 edges]>],
 '03': [<Graph [178 nodes, 409 edges]>],
 '07': [<Graph [107 nodes, 245 edges]>],
 '05': [<Graph [128 nodes, 310 edges]>],
 '04': [<Graph [162 nodes, 381 edges]>],
```

```
 '02': [<Graph [195 nodes, 497 edges]>],
 '09': [<Graph [140 nodes, 310 edges]>],
 '06': [<Graph [157 nodes, 396 edges]>],
 '08': [<Graph [143 nodes, 371 edges]>]}
```

Arizona has some missing edges in its graph

[11]:
```
graph.add_edge(208,378)
graph.add_edge(208,202)
graph.add_edge(208,780)
graph.add_edge(208,90)
graph.add_edge(208,42)
```

[12]:
```
graph
```

[12]:
```
<Graph [1489 nodes, 4097 edges]>
```

Recreate the initial partition

[13]:
```
contiguous_components(initial_partition)
```

[13]:
```
{'01': [<Graph [279 nodes, 724 edges]>],
 '03': [<Graph [178 nodes, 409 edges]>],
 '07': [<Graph [107 nodes, 245 edges]>],
 '05': [<Graph [128 nodes, 310 edges]>],
 '04': [<Graph [162 nodes, 381 edges]>],
 '02': [<Graph [195 nodes, 497 edges]>],
 '09': [<Graph [140 nodes, 310 edges]>],
 '06': [<Graph [157 nodes, 396 edges]>],
 '08': [<Graph [143 nodes, 371 edges]>]}
```

[14]:
```
election = Election("PRES16", {"Dem": "PRES16D", "Rep": "PRES16R"})

initial_partition = GeographicPartition(
    graph,
    assignment="CD",
    updaters={
        "cut_edges": cut_edges,
        "population": Tally("TOTPOP", alias="population"),
        "PRES16": election
    }
)
```

[15]:
```
for district, pop in initial_partition["population"].items():
    print("District {}: {}".format(district, pop))
```

```
District 01: 710176.0
District 03: 710252.0
District 07: 710220.0
```

```
District 05: 710224.0
District 04: 710224.0
District 02: 710244.0
District 09: 710224.0
District 06: 710221.0
District 08: 710232.0
```

[16]:
```
sum_population = sum(initial_partition["population"].values())
ideal_population = sum_population / len(initial_partition)

# We use functools.partial to bind the extra parameters (pop_col, pop_target,␣
 ↪epsilon, node_repeats)
# of the recom proposal.
proposal = partial(recom,
                   pop_col="TOTPOP",
                   pop_target=ideal_population,
                   epsilon=.05,
                   node_repeats=2
                  )
```

[17]:
```
compactness_bound = constraints.UpperBound(
    lambda p: len(p["cut_edges"]),
    2*len(initial_partition["cut_edges"])
)

pop_constraint = constraints.
 ↪within_percent_of_ideal_population(initial_partition, .05)
```

[18]:
```
def district_diff(partition1, partition2):
    percentage_change = []
    for (district1, graph1), (district2, graph2) in␣
 ↪zip(contiguous_components(partition1).items(),␣
 ↪contiguous_components(partition2).items()):
        if district1 == district2:
            set1 = set(graph1[0].nodes)
            set2 = set(graph2[0].nodes)
            if set1 != set2:
                set_diff1 = set1 - set2
                set_diff2 = set2 - set1
                set_intersection = set1 & set2
                diff = len(set_intersection)/len(set1)
                if diff > 1:
                    percentage_change.append(0)
                else:
                    percentage_change.append(diff)
            else:
                percentage_change.append(1)
```

```
    return percentage_change
```

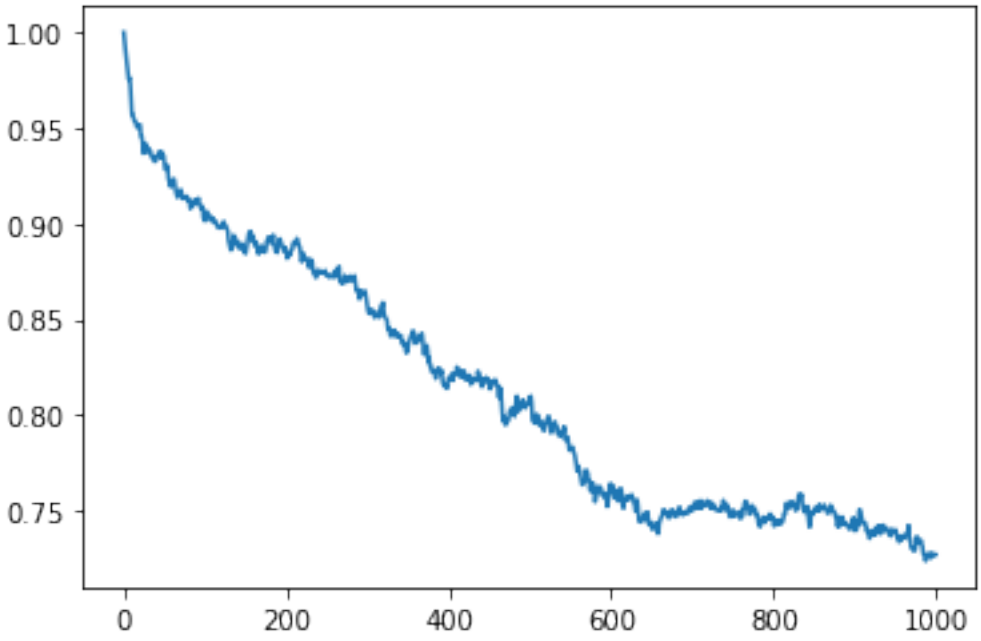
[19]:
```
from gerrychain import MarkovChain
from gerrychain.constraints import single_flip_contiguous, contiguous
from gerrychain.proposals import propose_random_flip
from gerrychain.accept import always_accept
steps = 1000
chain = MarkovChain(
    proposal=proposal,
    constraints=[single_flip_contiguous],
    accept=always_accept,
    initial_state=initial_partition,
    total_steps=steps
)
```

[20]:
```
last1 = None
best_partition = None
best_partition_similarity = 1
district_percent_change_per_partition = []
for partition in chain.with_progress_bar():
    district_differences = district_diff(initial_partition, partition)
    district_percent_change_per_partition.append(district_differences)
    last1 = partition
    partition_similarity = np.mean(district_differences)
    if best_partition_similarity > partition_similarity:
        best_partition_similarity = partition_similarity
        best_partition = partition
```

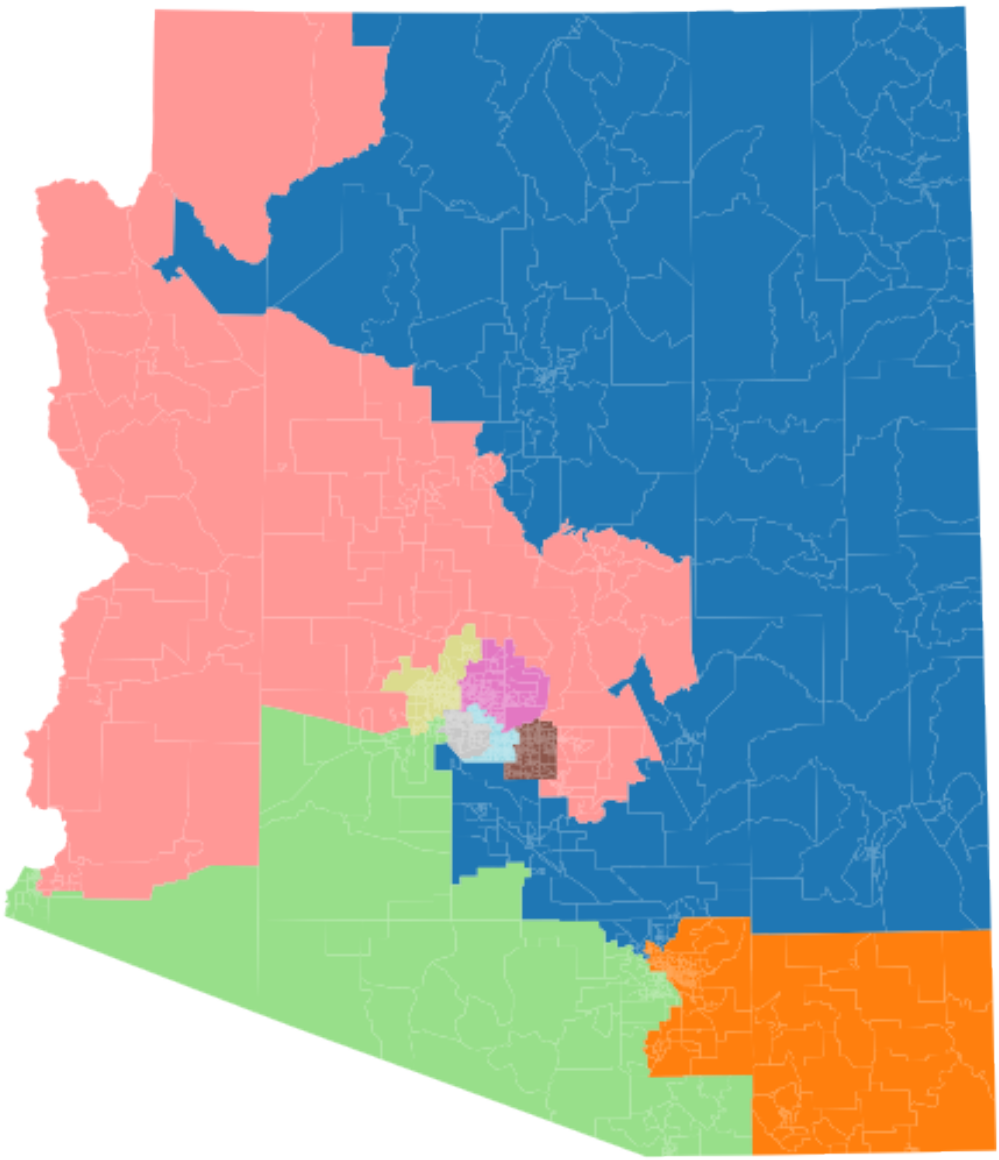
```
100%|      | 1000/1000 [14:41:02<00:00, 52.86s/it]
```

[21]:
```
import matplotlib.pyplot as plt
y = np.mean(district_percent_change_per_partition, axis=1)
plt.plot(y)
```

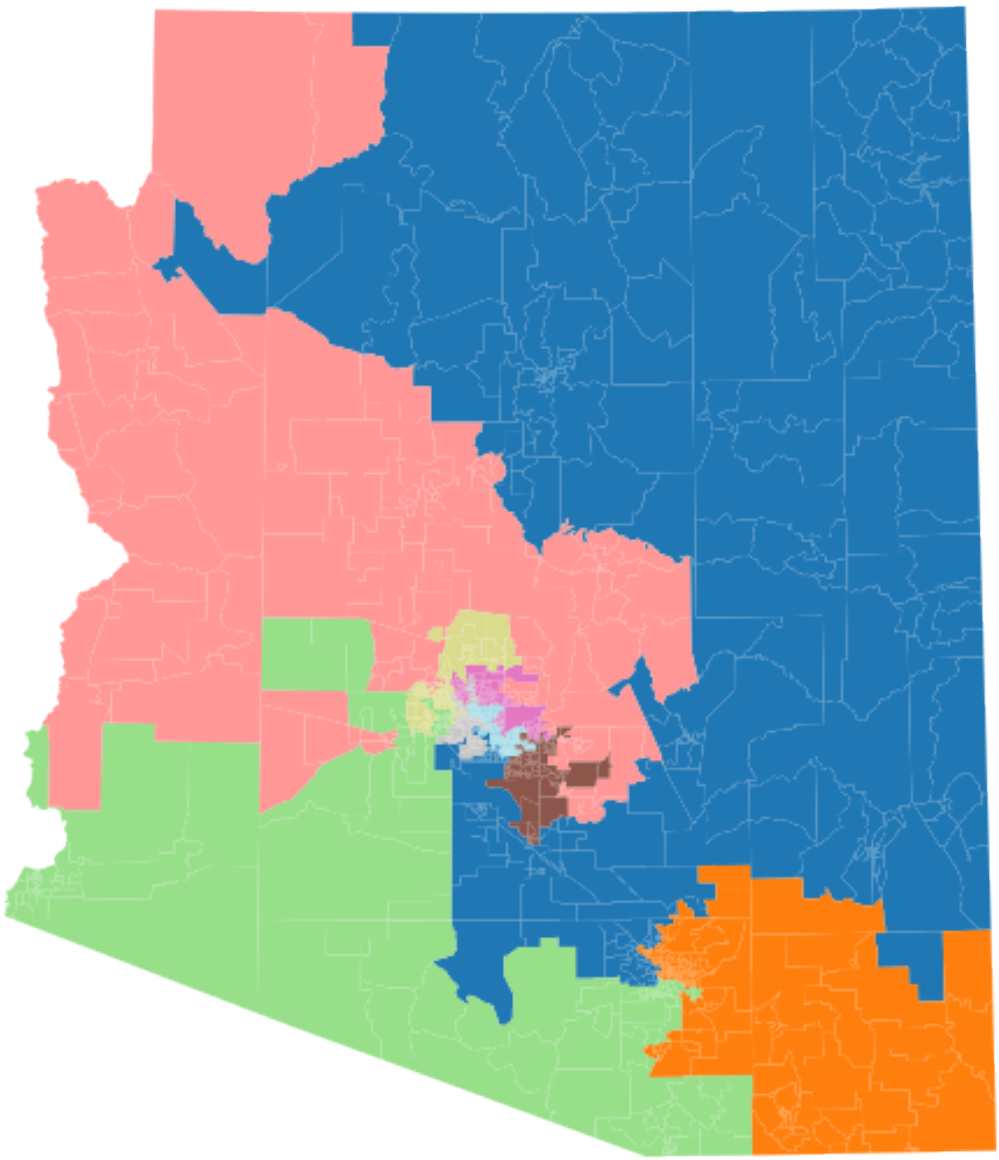
[21]:
```
[<matplotlib.lines.Line2D at 0x21c54b89c70>]
```



[22]:
```
initial_partition.plot(figsize=(10, 10), cmap="tab20")
plt.axis('off')
plt.show()
```



[23]:
```
best_partition.plot(figsize=(10, 10), cmap="tab20")
plt.axis('off')
plt.show()
```



[24]:
```
last1.plot(figsize=(10, 10), cmap="tab20")
plt.axis('off')
plt.show()
```