# Choropleth Mapping (PRACTICE)[¶](#Choropleth-Mapping-(PRACTICE))

Chroplet mapping allow us to display non-geographic attributes or variables on a geographic map. The word choropleth stems from the root “choro”, meaning “region”

In [1]:

%matplotlib inline

import seaborn

import pandas

import geopandas

import pysal

import numpy

import mapclassify

import matplotlib.pyplot as plt

In [2]:

fp = 'D:\Research\PROJECT\pyexeriences\Geospatial experiences\data\Shapefiles\sen\_admbnda\_adm2\_1m\_gov\_ocha\_20190426.shp'

In [3]:

mx = geopandas.read\_file(fp)

In [4]:

mx.head()

Out[4]:

|  | **Shape\_Leng** | **Shape\_Area** | **ADM2\_FR** | **ADM2\_PCODE** | **ADM2\_REF** | **ADM2ALT1FR** | **ADM2ALT2FR** | **ADM1\_FR** | **ADM1\_PCODE** | **ADM0\_FR** | **ADM0\_PCODE** | **date** | **validOn** | **validTo** | **geometry** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 5.352440 | 0.561518 | Bakel | SN1201 | None | None | None | Tambacounda | SN12 | Senegal | SN | 2017-08-04 | 2019-04-26 | None | POLYGON ((-12.66549 15.11117, -12.66378 15.111... |
| **1** | 1.453167 | 0.112038 | Bambey | SN0201 | None | None | None | Diourbel | SN02 | Senegal | SN | 2017-08-04 | 2019-04-26 | None | POLYGON ((-16.41398 15.02491, -16.40452 15.023... |
| **2** | 3.120739 | 0.443401 | Bignona | SN1401 | None | None | None | Ziguinchor | SN14 | Senegal | SN | 2017-08-04 | 2019-04-26 | None | POLYGON ((-15.89499 13.16475, -15.89556 13.162... |
| **3** | 1.513093 | 0.096306 | Birkelane | SN0401 | None | None | None | Kaffrine | SN04 | Senegal | SN | 2017-08-04 | 2019-04-26 | None | POLYGON ((-15.59026 14.11543, -15.58989 14.114... |
| **4** | 2.545793 | 0.238640 | Bounkiling | SN1101 | None | None | None | Sedhiou | SN11 | Senegal | SN | 2017-08-04 | 2019-04-26 | None | POLYGON ((-15.49857 13.39529, -15.49636 13.395... |

In [5]:

mx.columns

Out[5]:

Index(['Shape\_Leng', 'Shape\_Area', 'ADM2\_FR', 'ADM2\_PCODE', 'ADM2\_REF',

'ADM2ALT1FR', 'ADM2ALT2FR', 'ADM1\_FR', 'ADM1\_PCODE', 'ADM0\_FR',

'ADM0\_PCODE', 'date', 'validOn', 'validTo', 'geometry'],

dtype='object')

In [6]:

mx[['ADM1\_FR', 'Shape\_Area']].head()

Out[6]:

|  | **ADM1\_FR** | **Shape\_Area** |
| --- | --- | --- |
| **0** | Tambacounda | 0.561518 |
| **1** | Diourbel | 0.112038 |
| **2** | Ziguinchor | 0.443401 |
| **3** | Kaffrine | 0.096306 |
| **4** | Sedhiou | 0.238640 |

In [7]:

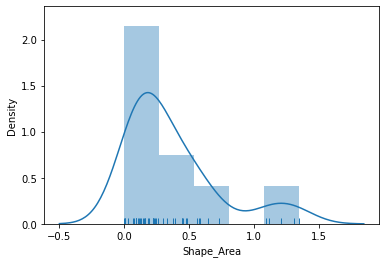
h = seaborn.distplot(mx['Shape\_Area'], bins=5, rug=True);

d:\programm files\python 3 8 6\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)

d:\programm files\python 3 8 6\lib\site-packages\seaborn\distributions.py:2055: FutureWarning: The `axis` variable is no longer used and will be removed. Instead, assign variables directly to `x` or `y`.

warnings.warn(msg, FutureWarning)



As we can see, the distribution is positively skewed as in common in regional shpae studies. In other words, the mean exceeds the median (50%, in the table above), leading the to fat right tail in the figure. As we shall see, this skewness will have implications for the choice of choropleth classification scheme.

In [8]:

mx['Shape\_Area'].describe()

Out[8]:

count 45.000000

mean 0.366317

std 0.355617

min 0.001186

25% 0.121879

50% 0.228846

75% 0.478131

max 1.348348

Name: Shape\_Area, dtype: float64

The matplotlib hist function under the hood to determine the class boundaries and the counts of observations in each class.

In [9]:

counts, bins, patches = h.hist(mx['Shape\_Area'], bins=5)

In [10]:

# The counts object captures how many observations each category in the classification has:

counts

Out[10]:

array([26., 9., 5., 0., 5.])

In [13]:

# The bin object stores these break points we are interested in when considering classification schemes

bins

Out[13]:

array([1.18575017e-03, 2.70618194e-01, 5.40050637e-01, 8.09483080e-01,

1.07891552e+00, 1.34834797e+00])

In [14]:

# The patches object can be ignored in this context, as it stores the geometries of the histogram plot

patches

Out[14]:

<BarContainer object of 5 artists>

In [15]:

# Equal interval

ei5 = mapclassify.EqualInterval(mx['Shape\_Area'], k=5)

ei5

Out[15]:

EqualInterval

Interval Count

--------------------

[0.00, 0.27] | 26

(0.27, 0.54] | 9

(0.54, 0.81] | 5

(0.81, 1.08] | 0

(1.08, 1.35] | 5

In [16]:

# Quantiles: To identify the class boundaries

q5 = mapclassify.Quantiles(mx.Shape\_Area, k=5)

q5

Out[16]:

Quantiles

Interval Count

--------------------

[0.00, 0.11] | 9

(0.11, 0.19] | 9

(0.19, 0.31] | 9

(0.31, 0.56] | 9

(0.56, 1.35] | 9

The number of values in each class are equal as well as the widths

In [17]:

q5.bins[1:]-q5.bins[:-1]

Out[17]:

array([0.08472227, 0.12179117, 0.25224095, 0.7838127 ])

In [18]:

# Mean-standard deviation: This classifier is best used when data is normally distributed or, at least, when the sample mean is a meaningful measure to anchor the classification around

msd = mapclassify.StdMean(mx['Shape\_Area'])

msd

Out[18]:

StdMean

Interval Count

----------------------

( -inf, -0.34] | 0

(-0.34, 0.01] | 3

( 0.01, 0.72] | 36

( 0.72, 1.08] | 1

( 1.08, 1.35] | 5

In [19]:

# Maximum Breaks: The maximum breaks classifier decides where to set the break points between classes by considering the difference between sorted values.

mb5 = mapclassify.MaximumBreaks(mx['Shape\_Area'], k=5)

mb5

Out[19]:

MaximumBreaks

Interval Count

--------------------

[0.00, 0.69] | 39

(0.69, 0.91] | 1

(0.91, 1.16] | 2

(1.16, 1.26] | 1

(1.26, 1.35] | 2

In [20]:

# Box-Plot: The box-plot classification is a blend of the quantile and standard deviation classifiers.

bp = mapclassify.BoxPlot(mx['Shape\_Area'])

bp

Out[20]:

BoxPlot

Interval Count

----------------------

( -inf, -0.41] | 0

(-0.41, 0.12] | 12

( 0.12, 0.23] | 11

( 0.23, 0.48] | 11

( 0.48, 1.01] | 6

( 1.01, 1.35] | 5

In [22]:

f, ax = plt.subplots(1, figsize=(9, 9))

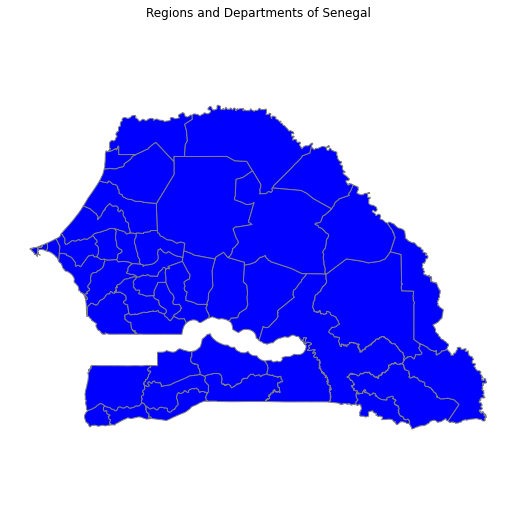
mx.plot(ax=ax, color='blue', edgecolor='grey')

ax.set\_axis\_off()

ax.set\_title('Regions and Departments of Senegal')

plt.axis('equal')

plt.show()



In [23]:

f, ax = plt.subplots(1, figsize=(9, 9))

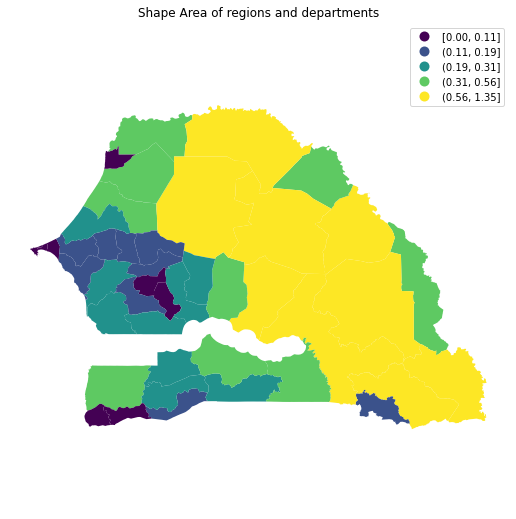
mx.plot(ax=ax, column='Shape\_Area', legend=True, scheme='Quantiles')

ax.set\_axis\_off()

ax.set\_title('Shape Area of regions and departments')

plt.axis('equal')

plt.show()



In [25]:

f, ax = plt.subplots(1, figsize=(9, 9))

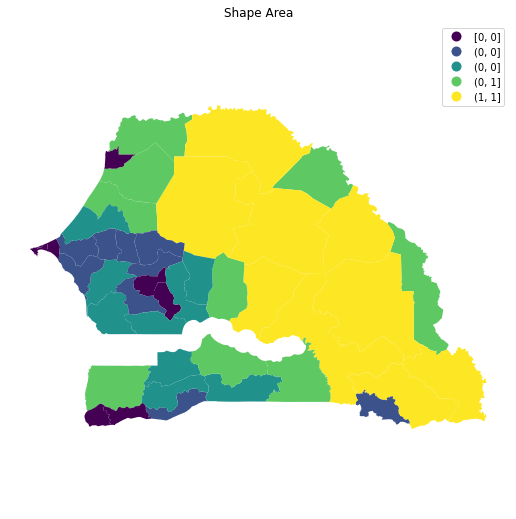
mx.plot(ax=ax, column='Shape\_Area', legend=True, scheme='Quantiles', legend\_kwds={'fmt':'{:.0f}'})

ax.set\_axis\_off()

ax.set\_title('Shape Area')

plt.axis('equal')

plt.show()



In [29]:

f, ax = plt.subplots(1, figsize=(9, 9))

mx.plot(ax=ax, column='ADM2\_FR', legend=True, scheme='Quantiles', legend\_kwds={'fmt':'{:.0f}'}, \

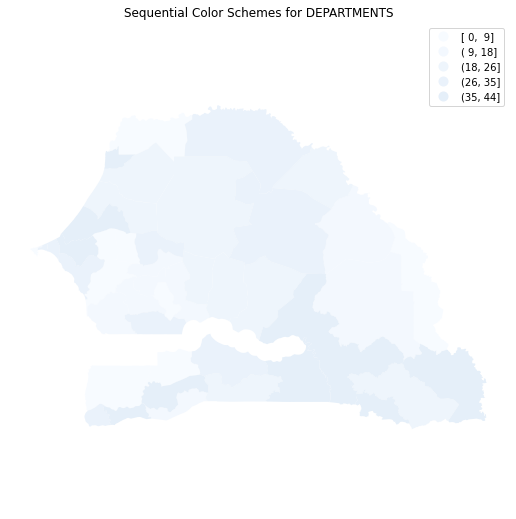
cmap='Blues')

ax.set\_axis\_off()

ax.set\_title('Sequential Color Schemes for DEPARTMENTS')

plt.axis('equal')

plt.show()



In [30]:

# Sequential color map with lighter shade

f, ax = plt.subplots(1, figsize=(9, 9))

mx.plot(ax=ax, column='ADM1\_FR', legend=True, scheme='Quantiles', legend\_kwds={'fmt':'{:.0f}'}, \

cmap='Blues', edgecolor='k')

ax.set\_axis\_off()

ax.set\_title('PCGDP1940')

plt.axis('equal')

plt.show()

