# **作业3**

马一清14300180095

注意：所有代码均可用notepad打开，并在python-spyder以及processing3(作业要求用Processing3)(~=java)(附件里已包含了processing3)环境下实现。

代码除了在word内出现外，还在附件里都包含一份完整版。

Question1:

Use python basemap/ GMAPS to visualize (demonstrate) the map of China using 3 different projection methods, i.e.: conformal, equal area, equal distance (Azimuthal) projections.

lcc 兰伯特保形投影 [Lambert Conformal]

保形映射，真实地再现原始球体的所有特征，假想地图投影将是完全[等距的](http://www.progonos.com/furuti/MapProj/Dither/CartProp/DistPres/distPres.html)，即每两个点之间的距离 将在地图和球体上保持相同的比率; 因此，所有形状也将被保留。

飞行员使用基于LCC的航空图，因为在Lambert保形圆锥投影上绘制的直线近似于典型飞行距离的端点之间的大圆路径。

Aea [Albers等面积投影](https://matplotlib.org/basemap/users/aea.html) [Albers Equal Area]

所述亚尔勃斯投影，或亚尔勃斯投影是一个圆锥，一种等面积地图投影，在1805年发表，虽然尺度和形状不被保留，但在标准平行线之间的扭曲是最小的。

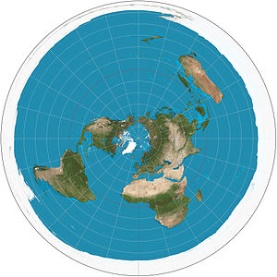
阿尔伯斯投影用于一个不列颠哥伦比亚省，是政府使用的唯一标准投影。它也被美国地质调查局和美国人口普查局使用。

|  |
| --- |
|  |

npaeqd 方位等距投影 [north-Polar Azimuthal Equidistant]

它具有地图上所有点与中心点成比例且距离正确，并且地图上的所有点都位于距中心点正确的方位角（方向）。它的应用是极地投影，其显示所有经线（经度线）为直线，距极点的距离表示正确。

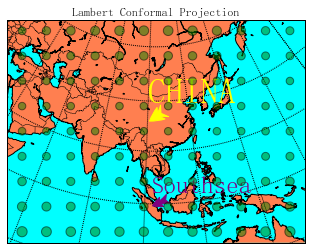
这个投影出现在许多文艺复兴时期的地图上，而杰拉德· [墨卡托（Gerardus Mercator）则](https://en.wikipedia.org/wiki/Gerardus_Mercator)用它作为北极地区的插图。



**代码1：**

|  |
| --- |
| from mpl\_toolkits.basemap import Basemap  import numpy as np  import matplotlib.pyplot as plt  # setup lambert conformal basemap.  # lat\_1 is first standard parallel.  # lat\_2 is second standard parallel (defaults to lat\_1).  # lon\_0,lat\_0 is central point.  # rsphere=(6378137.00,6356752.3142) specifies WGS4 ellipsoid  # area\_thresh=1000 means don't plot coastline features less  # than 1000 km^2 in area.  m = Basemap(width=12000000,height=9000000,  rsphere=(6378137.00,6356752.3142),\  resolution='l',area\_thresh=1000.,projection='lcc',\  lat\_1=53.,lat\_2=3,lat\_0=28,lon\_0=104.)  m.drawcoastlines()  m.drawcountries()  m.fillcontinents(color='coral',lake\_color='aqua')  # draw parallels and meridians.  m.drawparallels(np.arange(-80.,81.,20.))  m.drawmeridians(np.arange(-180.,181.,20.))  m.drawmapboundary(fill\_color='aqua')  # draw tissot's indicatrix to show distortion.  ax = plt.gca()  for y in np.linspace(m.ymax/20,19\*m.ymax/20,9):  for x in np.linspace(m.xmax/20,19\*m.xmax/20,12):  lon, lat = m(x,y,inverse=True)  poly = m.tissot(lon,lat,1.5,100,\  facecolor='green',zorder=10,alpha=0.5)  plt.title("Lambert Conformal Projection")  plt.show() |

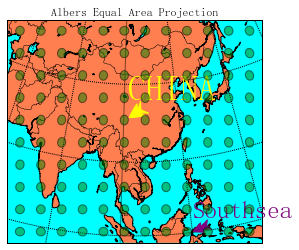
**效果图1：**

地图上的绿色圆形是地球表面上的等面积圆，可以用于显示地图投影的角度和面积变形，在保形投影上，圆形形状被保留。****

**代码2：**

|  |
| --- |
| from mpl\_toolkits.basemap import Basemap  import numpy as np  import matplotlib.pyplot as plt  # setup albers equal area conic basemap  # lat\_1 is first standard parallel.  # lat\_2 is second standard parallel.  # lon\_0,lat\_0 is central point.  m = Basemap(width=8000000,height=7000000,  resolution='l',projection='aea',\  lat\_1=53.,lat\_2=3,lon\_0=104,lat\_0=28)  #需要填入4个参数lat 1 lat2 lat\_0 lat\_0前两个是图的上维度，和下维度，后两个是中点的经纬度。中国领土的四端为：最东端（135°2′30’’E），最西端（73°29'59.79"E），最南端（3°31‘00'N'，东经112°17’09”E），[南沙群岛](http://baike.baidu.com/item/%E5%8D%97%E6%B2%99%E7%BE%A4%E5%B2%9B)，最北端（53°33′N，124°20′E）  求南北端均值后取整得到53,3,104=(135+73)/2,28=(53+3)/2  m.drawcoastlines()  m.drawcountries()  m.fillcontinents(color='coral',lake\_color='aqua')  # draw parallels and meridians.  m.drawparallels(np.arange(-80.,81.,20.))  m.drawmeridians(np.arange(-180.,181.,20.))  m.drawmapboundary(fill\_color='aqua')  # draw tissot's indicatrix to show distortion.  ax = plt.gca()  for y in np.linspace(m.ymax/20,19\*m.ymax/20,10):  for x in np.linspace(m.xmax/20,19\*m.xmax/20,12):  lon, lat = m(x,y,inverse=True)  poly = m.tissot(lon,lat,1.25,100,\  facecolor='green',zorder=10,alpha=0.5)  plt.title("Albers Equal Area Projection")  China1\_lat=28  China1\_lon=104  x,y=map(China1\_lon-10,China1\_lat+5)  #设定标注格式  plt.annotate("CHINA",fontsize=35,xy=(x,y),xycoords='data',xytext=(0,20),textcoords='offset points',color='yellow',arrowprops=dict(arrowstyle='fancy',color='yellow'))  South\_lat=3  South\_lon=112  x,y=map(South\_lon,South\_lat)#设定标注格式  plt.annotate("Southsea" ,fontsize=25,xy=(x,y),xycoords='data',xytext=(0,15),textcoords='offset points',color='purple',arrowprops=dict(arrowstyle='fancy',color='purple'))  plt.show() |

**效果图2：**

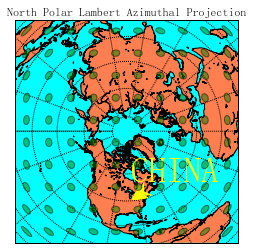


地图上的绿色圆形是地球表面上的等面积圆，可以用于显示地图投影的角度和面积变形。在等面积投影中，该区域保留但形状变了，可以看出极点附近的变形非常大。

**代码3：**

|  |
| --- |
| from mpl\_toolkits.basemap import Basemap  import numpy as np  import matplotlib.pyplot as plt  from matplotlib.patches import Polygon  # setup north polar lambert azimuthal basemap.  # The longitude lon\_0 is at 6-o'clock, and the  # latitude circle boundinglat is tangent to the edge  # of the map at lon\_0.  m = Basemap(projection='nplaea',boundinglat=5,lon\_0=270,resolution='l')  m.drawcoastlines()  m.fillcontinents(color='coral',lake\_color='aqua')  # draw parallels and meridians.  m.drawparallels(np.arange(-80.,81.,20.))  m.drawmeridians(np.arange(-180.,181.,20.))  m.drawmapboundary(fill\_color='aqua')  # draw tissot's indicatrix to show distortion.  ax = plt.gca()  for y in np.linspace(m.ymax/20,19\*m.ymax/20,10):  for x in np.linspace(m.xmax/20,19\*m.xmax/20,10):  lon, lat = m(x,y,inverse=True)  poly = m.tissot(lon,lat,2.5,100,\  facecolor='green',zorder=10,alpha=0.5)  plt.title("North Polar Lambert Azimuthal Projection")  China1\_lat=28  China1\_lon=140  x,y=map(China1\_lon,China1\_lat)  #设定标注格式  plt.annotate("CHINA",fontsize=35,xy=(x,y),xycoords='data',xytext=(0,20),textcoords='offset points',color='yellow',arrowprops=dict(arrowstyle='fancy',color='yellow'))  plt.show() |

**效果图3：**

****

Question2:

**P159 6.3**

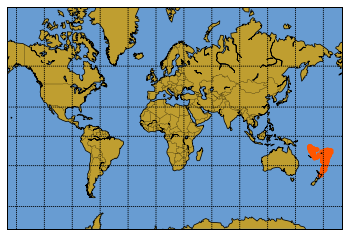
**代码1：（在R中保存数据集）**

|  |
| --- |
| View(quakes)  write.csv(quakes, file="C:/Users/Air/Desktop/quakes.csv") |

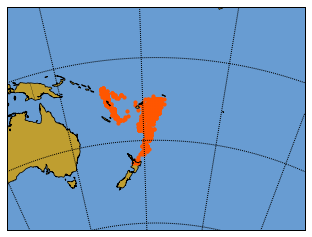
**代码2：采用eqdc等距圆锥投影**

|  |
| --- |
| # -\*- coding: utf-8 -\*-  """  Created on Thu Apr 27 11:20:00 2017  @author: Air  """  import pandas as pd  from mpl\_toolkits.basemap import Basemap  import numpy as np  import matplotlib.pyplot as plt  #============================================# read data  quakes=pd.read\_csv("quakes.csv")  lat=[]  long=[]  depth=[]  mag=[]  stations=[]  for i in range(999):  if quakes.iat[i,4]>5.5:  lat.append(int(quakes.iat[i,1]))  long.append(int(quakes.iat[i,2]))  depth.append(int(quakes.iat[i,3]))  mag.append(int(quakes.iat[i,4]))  stations.append(int(quakes.iat[i,5]))  #============================================  m=Basemap(width=12000000,height=9000000,resolution='l',projection='eqdc',lat\_1=-20,lat\_2=-24,lat\_0=-22,lon\_0=185)  m.drawcoastlines(linewidth=0.75)  m.drawcountries(linewidth=0.25)  m.drawmapboundary(fill\_color='#689CD2')  m.fillcontinents(color='#BF9E30',lake\_color='#689CD2',zorder=0)  m.drawmeridians(np.arange(0,360,30))  m.drawparallels(np.arange(-90,90,30))  #============================================  x,y=m(long,lat)  max\_depth=max(depth)  #============================================  size\_factor=100  y\_offset=15  rotation=-30  for i,j,d,ma in zip(x,y,depth,mag):  cs=m.scatter(i,j,s=10,marker='o',color='#FF5600')  plt.text(i,j+y\_offset,repr(ma),rotation=rotation,fontsize=10)  plt.show() |

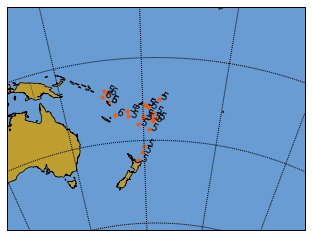
**效果图：**



一开始使用的是墨卡托投影，标注了1000个点.

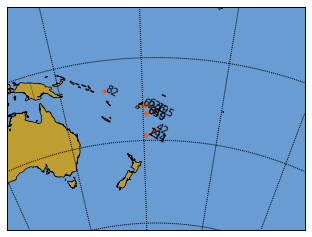
****

发现了几乎所有的点都集中在一块，于是改用eqdc投影，通过改变投影映射的经纬度，可以局部放大地震集中的位置.

****

通过if 条件语句，可以显示震级大于5.5的点的位置.

用plt.text实现数字标识.



可以更改方式，标注深度depth.

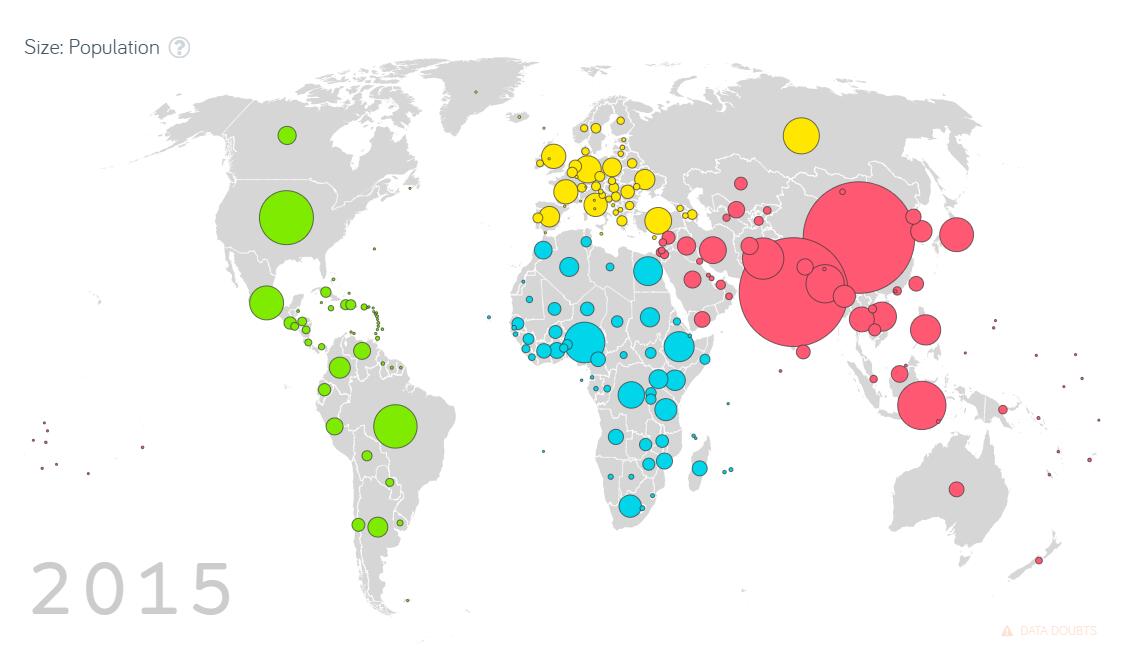
综上，实现了题目中要求的标注数据。

## Question3:

P159 6.4

首先用google chorme 浏览器提供的SSLspeedy插件进行翻墙，然后下载人均GDP数据。

我希望画的图如[www.gapminder.org](http://www.gapminder.org)上的一样。所以首先在python\_spyder里面编写地图投影的一张张图，然后到processing3里面实现点的移动。



数据预处理：做了以下四件事

1．首先，为了在Python里用basemap画出图片，我首先整理GDP数据，得到了各国的大陆属性label(欧，亚，非，拉，美，澳，南极)，这是为了方便给数据点标注不同颜色

2．数据人均GDP的EXCEL表格里面包含了1960-2011年的人均GDP数据，我们选取最后20年1991-2011年的数据进行绘图。

3．将人均GDP数据中数据有缺失的国家删除了，这样保证每个国家的20年人均GDP完整的国家有164个。

4．原始数据集只有国家的名称和代号【Australia】【AU】没有经纬度，所以我从【<https://developers.google.com/public-data/docs/canonical/countries_csv>】网站下载对应各国的经纬度数据，整合到GDP\_excel表格中。这是为了在地图上标注点。

最后得到GDP\_perfect.csv文档，再导入Python中作图.

## 数据绘图：

想变成不一样的颜色，希望不同大洲的岛屿变成不同的颜色，所以用了landpolygans的信息，进行不同的填色

|  |
| --- |
| polys = []  for polygon in m.landpolygons:  polys.append(polygon.get\_coords())  lc = PolyCollection(polys[0:1],edgecolor='salmon',facecolor='salmon', closed=False)  plt.gca().add\_collection(lc)  lc = PolyCollection(polys[1:2],edgecolor='lightpink',facecolor='lightpink', closed=False)  plt.gca().add\_collection(lc) |

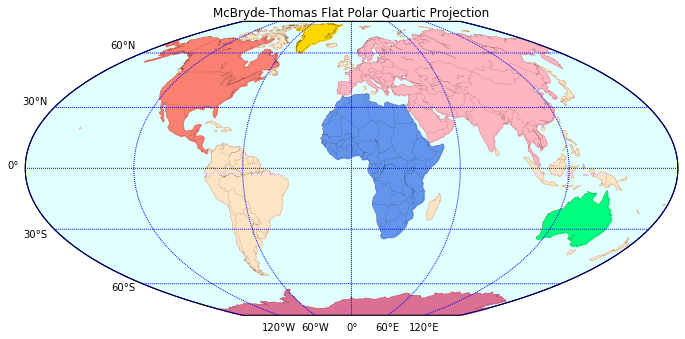
然后通过改变fig的属性调整图片大小

|  |
| --- |
| fig = plt.figure(figsize=(11.7,8.3)) |

然后通过map.draparallels添加label

|  |
| --- |
| m.drawmeridians(np.arange(0,360,60),labels=[0,0,0,1],color='blue')  m.drawparallels(np.arange(-90,90,30),labels=[1,0,0,0],color='blue') |

现在暂时画出的图如：

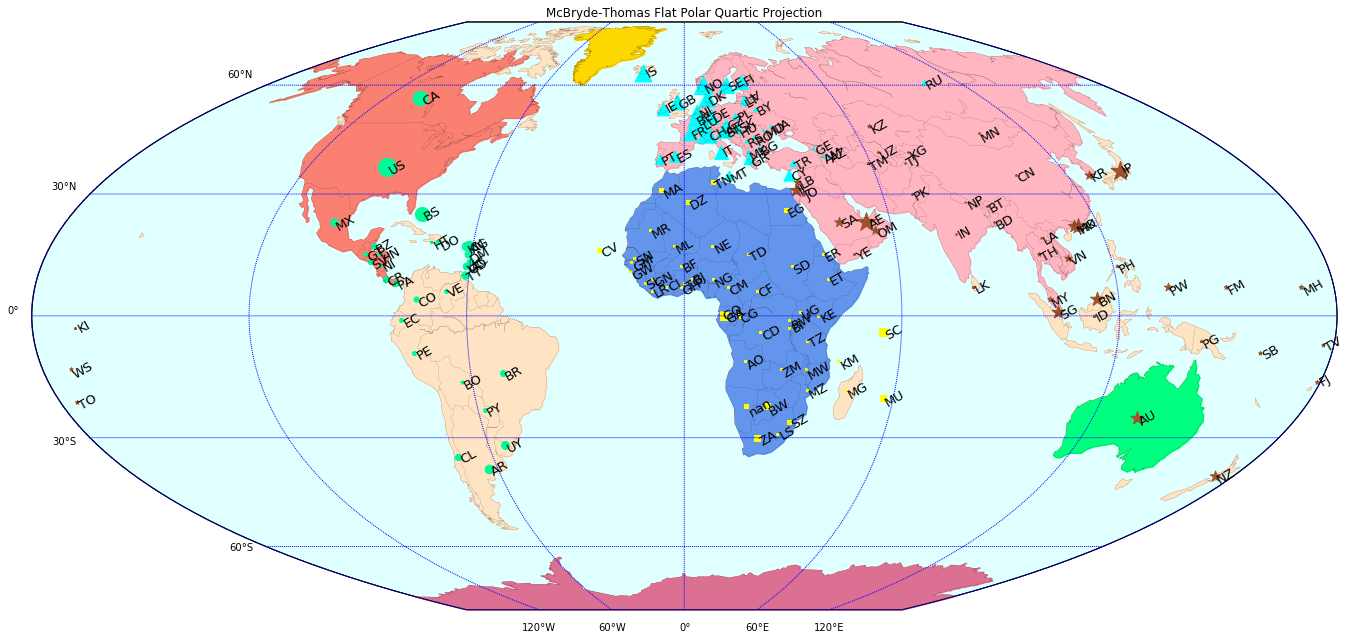


然后为数据添加标注点：添加gpa的点。

|  |
| --- |
| y\_offset=15  rotation=-30  size\_factor = 200.0  for i,j,g,c,e in zip(x,y,group,id,economic1992):  if g=="[Africa]":  size = size\_factor\*e/me  cs=m.scatter(i,j,s=size,marker='o',color='yellow')  plt.text(i,j+y\_offset,c,rotation=-rotation,fontsize=8)  if g=="[Asia]":  size = size\_factor\*e/me  cs=m.scatter(i,j,s=size,marker='o',color='sienna')  plt.text(i,j+y\_offset,c,rotation=-rotation,fontsize=8)  if g=="[Europe]":  size = size\_factor\*e/me  cs=m.scatter(i,j,s=size,marker='o',color='aqua')  plt.text(i,j+y\_offset,c,rotation=-rotation,fontsize=8)  if g=="[America]":  size = size\_factor\*e/me  cs=m.scatter(i,j,s=size,marker='o',color='mediumspringgreen')  plt.text(i,j+y\_offset,c,rotation=-rotation,fontsize=8) |

像这段代码中一样，为不同大洲的各国的数据加了经济标注，对于非洲是s正方形，对于欧洲是^三角形，对于亚洲澳洲是\*五角星,对于美洲是o圆圈

用点的大小表示人均GDP的高低，用点上面的标注名称表示国家代号，对1992年的经济数据先做一个图。

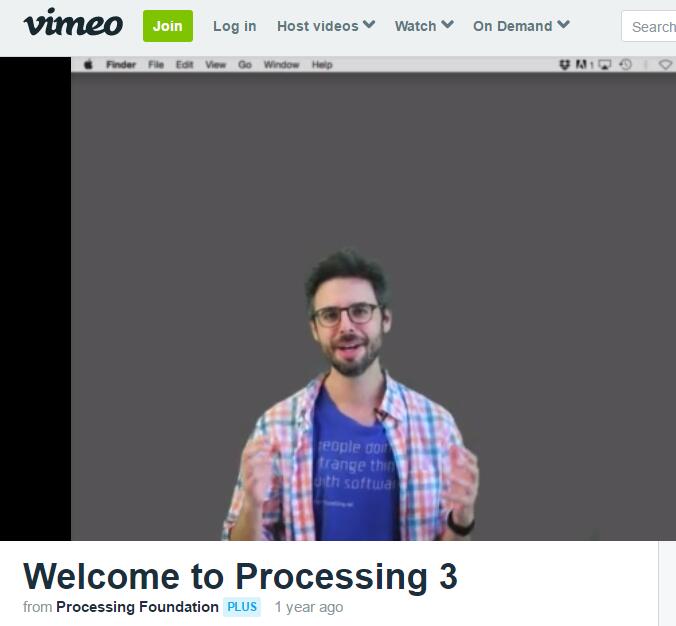
得到的结果如下：  
然后我要用Processing实现数据的动态化。

思路1.0：

本来是想用上图在processing3里实现数据点的移动的，后来发现不太可行，因为要从这张图中分离出数据点，需要在processing3里用JAVA重现一遍。于是换了一种思路

## 思路2.0

在processing3重现地图和地图点，这款软件是配套java的，所以我学习了一段时间的Java之后，看了processing3的教程



我做了以下4件事：

1. 首先是用<https://en.wikipedia.org/wiki/File:World_map_-_low_resolution.svg>下载世界地图的svg，（形状格式）这样可以用getchild函数实现了全世界地图各个地区显示，并且着色。

|  |
| --- |
| world = loadShape("World\_map.svg");  me= world.getChild(3);  me.disableStyle();  fill(0, 197, 205);  noStroke();  shape(me, -2, 1); |

2. 然后我希望能通过小球形状，大小来表现具体的GDP大小，用了java中的sin 和ellipse函数实现。

|  |
| --- |
| float ang1= radians(angle1);  x1=height\*0.42+(scalar[0+i\*13]\*sin(ang1));  fill(255, 228, 225);  ellipse(width\*0.305- 120,x1, scalar[+i\*13], scalar[0+i\*13]);  angle1 +=speed[0+i\*13]; |

3.需要重新整理一遍数据：

因为svg格式的图只能通过输入国家的编号得到，所以通过查表，我挑出了以下13个城市

希望实现其五年GDP的变化的动态图（2007-2011）

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Entity** | **number** | Lat | long | 2007 | 2008 | 2009 | 2010 | 2011 |
| Mexico | 3 | 23.6345 | -102.553 | 6333.0824 | 6327.0132 | 5875.62 | 6124.7096 | 6288.2532 |
| Algeria | 5 | 28.03389 | 1.659626 | 2155.4852 | 2173.7879 | 2192.704 | 2231.9802 | 2255.2255 |
| Egypt | 25 | 26.82055 | 30.8025 | 1765.8694 | 1858.8584 | 1911.9645 | 1975.55 | 1976.6148 |
| Congo, Dem. Rep. | 44 | -4.03833 | 21.75866 | 97.910183 | 101.10385 | 101.16901 | 105.53174 | 109.8093 |
| Japan | 113 | 36.20482 | 138.2529 | 40837.267 | 40433.001 | 38242.024 | 39971.787 | 39578.074 |
| Madagascar | 141 | -18.7669 | 46.86911 | 255.09415 | 265.36636 | 245.94615 | 242.68582 | 238.17275 |
| Canada | 227 | 56.13037 | -106.347 | 26229.743 | 26101.764 | 25069.869 | 25575.217 | 25933.288 |
| Russia | 254 | 61.52401 | 105.3188 | 2888.8474 | 3043.6656 | 2806.4148 | 2928.005 | 3054.7277 |
| Chile | 250 | -35.6751 | -71.543 | 6410.8067 | 6581.358 | 6451.6311 | 6781.6965 | 7122.9385 |
| China | 255 | 35.86166 | 104.1954 | 1864.1027 | 2032.615 | 2208.4039 | 2426.3325 | 2639.5416 |
| Turkey | 280 | 38.96375 | 35.24332 | 5323.683 | 5288.4161 | 4968.579 | 5356.0004 | 5741.4104 |
| Spain | 305 | 40.46367 | -3.74922 | 16351.111 | 16251.837 | 15523.003 | 15418.82 | 15428.321 |
| Italy | 325 | 41.87194 | 12.56738 | 20291.227 | 19903.457 | 18697.22 | 18944.414 | 18937.25 |

同时，我把人均GDP数据进行了归一化处理，等比例转换成可以使用的速度和尺寸大小。得到：

数组1——小球尺寸

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 25.33233 | 25.30805 | 23.50248 | 24.49884 | 25.15301 |
| 8.621941 | 8.695152 | 8.770816 | 8.927921 | 9.020902 |
| 7.063478 | 7.435434 | 7.647858 | 7.9022 | 7.906459 |
| 0.391641 | 0.404415 | 0.404676 | 0.422127 | 0.439237 |
| 163.3491 | 161.732 | 152.9681 | 159.8871 | 158.3123 |
| 1.020377 | 1.061465 | 0.983785 | 0.970743 | 0.952691 |
| 104.919 | 104.4071 | 100.2795 | 102.3009 | 103.7332 |
| 11.55539 | 12.17466 | 11.22566 | 11.71202 | 12.21891 |
| 25.64323 | 26.32543 | 25.80652 | 27.12679 | 28.49175 |
| 7.456411 | 8.13046 | 8.833616 | 9.70533 | 10.55817 |
| 21.29473 | 21.15366 | 19.87432 | 21.424 | 22.96564 |
| 65.40444 | 65.00735 | 62.09201 | 61.67528 | 61.71328 |
| 81.16491 | 79.61383 | 74.78888 | 75.77766 | 75.749 |

数组2——速度

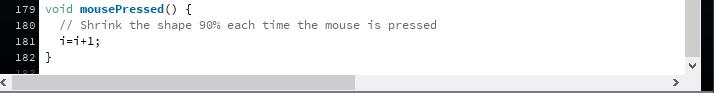
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1.899925 | 1.898104 | 1.762686 | 1.837413 | 1.886476 |
| 0.646646 | 0.652136 | 0.657811 | 0.669594 | 0.676568 |
| 0.529761 | 0.557658 | 0.573589 | 0.592665 | 0.592984 |
| 0.029373 | 0.030331 | 0.030351 | 0.03166 | 0.032943 |
| 12.25118 | 12.1299 | 11.47261 | 11.99154 | 11.87342 |
| 0.076528 | 0.07961 | 0.073784 | 0.072806 | 0.071452 |
| 7.868923 | 7.830529 | 7.520961 | 7.672565 | 7.779987 |
| 0.866654 | 0.9131 | 0.841924 | 0.878402 | 0.916418 |
| 1.923242 | 1.974407 | 1.935489 | 2.034509 | 2.136882 |
| 0.559231 | 0.609784 | 0.662521 | 0.7279 | 0.791862 |
| 1.597105 | 1.586525 | 1.490574 | 1.6068 | 1.722423 |
| 4.905333 | 4.875551 | 4.656901 | 4.625646 | 4.628496 |
| 6.087368 | 5.971037 | 5.609166 | 5.683324 | 5.681175 |

数组1是依据GDP高低得到的小球大小；数组2是依据GDP得来小球的震荡快慢。

4.Loop循环实现：

每一年我都可以看到13个城市不同的GDP大小，都用一个小球表示，小球越大，GDP越高，小球震荡越快,GDP越高

然后点击鼠标后，切换到下一年的GDP大小构成的小球。



总的processing3

## 代码：

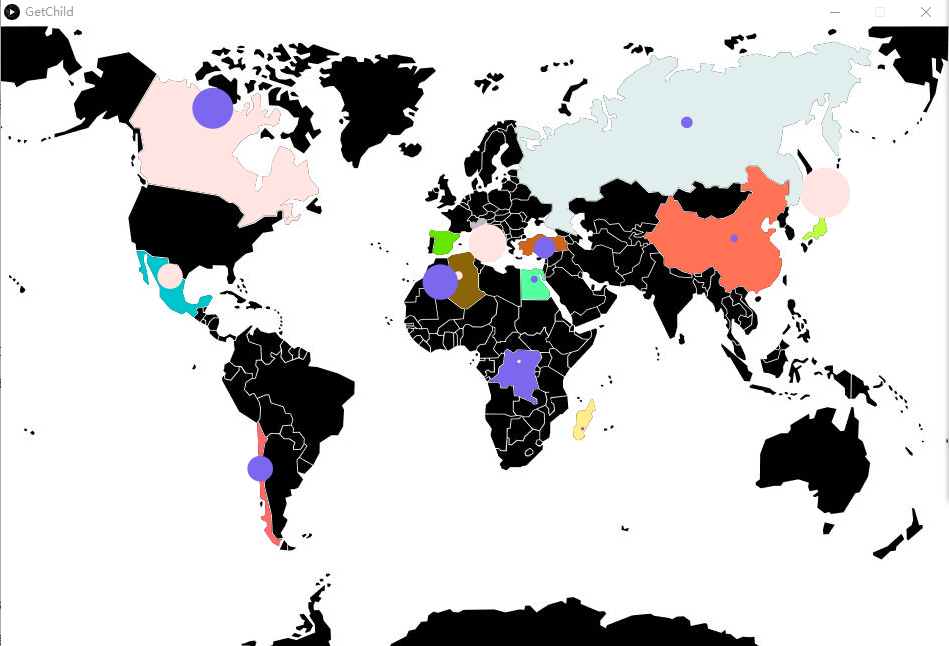
|  |
| --- |
| /\*\*  \* Get Child.  \*  \* SVG files can be made of many individual shapes.  \* Each of these shapes (called a "child") has its own name  \* that can be used to extract it from the "parent" file.  \* This example loads a map of the United States and creates  \* two new PShape objects by extracting the data from two states.  \*/  PImage bg;PShape world;PShape me;PShape al;PShape eg;PShape co;PShape jp;PShape ma;PShape ca;  PShape ru;PShape chile;PShape china;PShape tu;PShape sp;PShape it;  int i=0;  float x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13;  float angle1,angle2,angle3,angle4, angle5,angle6,angle7,angle8, angle9,angle10,angle11,angle12,angle13;  float []scalar={25.33232955,8.621940924,7.063477708,3.1640732,50.3490666,3.020376609,40.9189723,11.55538942  ,25.64322693,7.456410809,21.29473195,35.40444432,38.16490656,25.30805,8.695152,7.435434  ,0.404415,161.732,1.061465,104.4071,12.17466,26.32543,8.13046,21.15366,65.00735,79.61383  ,23.50248,8.770816,7.647858,0.404676,152.9681,0.983785,100.2795,11.22566,25.80652,8.833616,19.87432  ,62.09201,74.78888,24.49884,8.927921,7.9022,0.422127,159.8871,0.970743,102.3009,11.71202,27.12679  ,9.70533,21.424,61.67528,75.77766,25.15301,9.020902,7.906459,0.439237,158.3123,0.952691  ,103.7332,12.21891,28.49175,10.55817,22.96564,61.71328,75.749  };  float []speed={1.899924717,0.646645569,0.529760828,0.2373055,12.25117999,0.26528246,7.868922924,0.866654206,1.92324202,0.559230811,1.597104896,4.905333324,6.087367992,1.898104  ,0.652136,0.557658,0.030331,12.1299,0.07961,7.830529,0.9131,1.974407,0.609784,1.586525,4.875551,5.971037  ,1.762686,0.657811,0.573589,0.030351,11.47261,0.073784,7.520961,0.841924,1.935489,0.662521,1.490574  ,4.656901,5.609166,1.837413,0.669594,0.592665,0.03166,11.99154,0.072806,7.672565,0.878402,2.034509,0.7279  ,1.6068,4.625646,5.683324,1.886476,0.676568,0.592984,0.032943,11.87342,0.071452,7.779987,0.916418,2.136882  ,0.791862,1.722423,4.628496,5.681175  };  void setup() {  size(948, 621);  world = loadShape("World\_map.svg");  bg = loadImage("world-map.jpg");me= world.getChild(3);al= world.getChild(5);eg= world.getChild(25);co= world.getChild(44);  jp= world.getChild(113); ma= world.getChild(141);ca= world.getChild(227);ru= world.getChild(254);chile= world.getChild(250);  china= world.getChild(255);tu= world.getChild(280);sp= world.getChild(305); it= world.getChild(325);  rectMode(CENTER);  }  void draw() {    background(bg);  // Disable the colors in the SVG file  me.disableStyle();  fill(0, 197, 205);  noStroke();  shape(me, -2, 1);  //ball  float ang1= radians(angle1);  x1=height\*0.42+(scalar[0+i\*13]\*sin(ang1));  fill(255, 228, 225);  ellipse(width\*0.305- 120,x1, scalar[+i\*13], scalar[0+i\*13]);  angle1 +=speed[0+i\*13];  //  al.disableStyle();  fill(139, 101, 8);  noStroke();  shape(al, -2, 1);  //ball  float ang2= radians(angle2);  x2=height\*0.39+(scalar[1+i\*13]\*sin(ang2));  fill(255, 228, 225);  ellipse(width\*0.609- 120,x2, scalar[1+i\*13], scalar[1+i\*13]);  angle2 +=speed[1+i\*13];  //  eg.disableStyle();  fill(84, 255, 159);  noStroke();  shape(eg, -2, 1);  float ang3= radians(angle3);  x3=height\*0.41+(scalar[2+i\*13]\*sin(ang3));  fill(123,104, 238);  ellipse(width\*0.689- 120,x3, scalar[2+i\*13], scalar[2+i\*13]);  angle3 +=speed[2+i\*13];  //  co.disableStyle();  fill(123,104, 238);  noStroke();  shape(co, -2, 1);  float ang4= radians(angle4);  x4=height\*0.54+(scalar[3+i\*13]\*sin(ang4));  fill(255, 228, 225);  ellipse(width\*0.673- 120,x4, scalar[3+i\*13], scalar[3+i\*13]);  angle4 +=speed[3+i\*13];  //  jp.disableStyle();  fill(192, 255, 62);  noStroke();  shape(jp, -2, 1);  float ang5= radians(angle5);  x5=height\*0.285+(scalar[4+i\*13]\*sin(ang5));  fill(255, 228, 225);  ellipse(width\*0.98-105,x5, scalar[4+i\*13], scalar[4+i\*13]);  angle5 +=speed[4+i\*13];  //  ma.disableStyle();  fill(255, 236, 139);  noStroke();  shape(ma, -2, 1);  float ang6= radians(angle6);  x6=height\*0.65+(scalar[5+i\*13]\*sin(ang6));  fill(123,104, 238);  ellipse(width\*0.74-120,x6, scalar[5+i\*13], scalar[5+i\*13]);  angle6 +=speed[5+i\*13];  //  ca.disableStyle();  fill(255, 228, 225);  noStroke();  shape(ca, -2, 1);  float ang7= radians(angle7);  x7=height\*0.2+(scalar[6+i\*13]\*sin(ang7));  fill(123,104, 238);  ellipse(width\*0.35-120,x7, scalar[6+i\*13], scalar[6+i\*13]);  angle7 +=speed[6+i\*13];  //  ru.disableStyle();  fill(224, 238, 238);  noStroke();  shape(ru, -2, 1);  float ang8= radians(angle8);  x8=height\*0.17+(scalar[7+i\*13]\*sin(ang8));  fill(123,104, 238);  ellipse(width\*0.85-120,x8, scalar[7+i\*13], scalar[7+i\*13]);  angle8 +=speed[7+i\*13];  //  chile.disableStyle();  fill(255, 106, 106);  noStroke();  shape(chile, -2, 1);  float ang9= radians(angle9);  x9=height\*0.74+(scalar[8+i\*13]\*sin(ang9));  fill(123,104, 238);  ellipse(width\*0.4-120,x9, scalar[8+i\*13], scalar[8+i\*13]);  angle9 +=speed[8+i\*13];  //  china.disableStyle();  fill(255, 114, 86);  noStroke();  shape(china, -2, 1);  float ang10= radians(angle10);  x10=height\*0.34+(scalar[9+i\*13]\*sin(ang10));  fill(123,104, 238);  ellipse(width\*0.9-120,x10, scalar[9+i\*13], scalar[9+i\*13]);  angle10 +=speed[9+i\*13];  //  tu.disableStyle();  fill(205, 102, 29);  noStroke();  shape(tu, -2, 1);  float ang11= radians(angle11);  x11=height\*0.36+(scalar[10+i\*13]\*sin(ang11));  fill(123,104, 238);  ellipse(width\*0.7-120,x11, scalar[10+i\*13], scalar[10+i\*13]);  angle11 +=speed[10+i\*13];  //  sp.disableStyle();  fill(100, 231, 1);  noStroke();  shape(sp, -2, 1);  float ang12= radians(angle12);  x12=height\*0.36+(scalar[11+i\*13]\*sin(ang12));  fill(123,104, 238);  ellipse(width\*0.59-120,x12, scalar[11+i\*13], scalar[11+i\*13]);  angle12 +=speed[11+i\*13];  //  it.disableStyle();  fill(205,181, 205);  noStroke();  shape(it, -2, 1);  float ang13= radians(angle13);  x13=height\*0.32+(scalar[12+i\*13]\*sin(ang13));  fill(255, 228, 225);  ellipse(width\*0.64-120,x13, scalar[12+i\*13], scalar[12+i\*13]);  angle13 +=speed[12+i\*13];  }  void mousePressed() {  // Shrink the shape 90% each time the mouse is pressed  i=i+1;  } |

结果是个视频，放在3里的application.windows64的getChild里！

视频效果里有用手机拍摄的版本！

代码也在里面

图片效果如下。



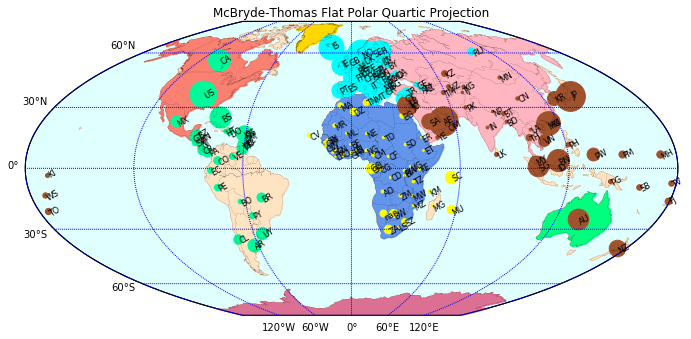
## Question4:

分析数据密集程度，那我用python spyder把上一题GDP所有的都点都标注到图上，看哪里的点最密集。

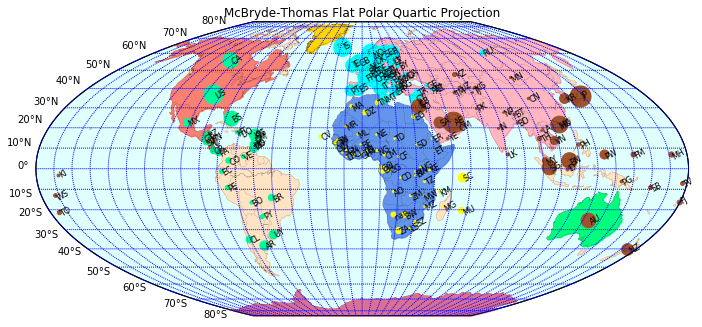
## 代码：

|  |
| --- |
| # -\*- coding: utf-8 -\*-  """  Created on Mon May 01 00:15:02 2017  @author: Air  """  # -\*- coding: utf-8 -\*-  """  Created on Sat Apr 29 23:14:11 2017  @author: Air  """  # -\*- coding: utf-8 -\*-  """  Created on Sat Apr 29 18:06:31 2017  @author: Air  """  import pandas as pd  from mpl\_toolkits.basemap import Basemap  import numpy as np  import matplotlib.pyplot as plt  from matplotlib.collections import PolyCollection  #============================================# read data  GDP=pd.read\_csv("GDP\_perfect.csv")  countries=[]  group=[]  id=[]  lat=[]  long=[]  economic1992=[]  economic1993=[]  #...  economic2011=[]  for i in range(161):  countries.append(GDP.iat[i,0])  group.append(GDP.iat[i,1])  id.append(GDP.iat[i,2])  lat.append(int(GDP.iat[i,3]))  long.append(int(GDP.iat[i,4]))  economic1992.append(int(GDP.iat[i,5]))  economic1993.append(int(GDP.iat[i,6]))  #...  economic2011.append(int(GDP.iat[i,24]))  #============================================  fig = plt.figure(figsize=(11.7,8.3))  m = Basemap(projection='mbtfpq',lon\_0=0,resolution='c')  m.drawcoastlines(linewidth=0.1)  m.drawcountries(linewidth=0.1)  m.drawmapboundary(fill\_color='lightcyan')  m.fillcontinents(color='coral',zorder=0)  m.drawmeridians(np.arange(0,360,60),labels=[0,0,0,1],color='blue')  m.drawparallels(np.arange(-90,90,30),labels=[1,0,0,0],color='blue')  #============================================  polys = []  for polygon in m.landpolygons:  polys.append(polygon.get\_coords())  lc = PolyCollection(polys[0:1],edgecolor='salmon',facecolor='salmon', closed=False)  plt.gca().add\_collection(lc)  lc = PolyCollection(polys[1:2],edgecolor='lightpink',facecolor='lightpink', closed=False)  plt.gca().add\_collection(lc)  lc = PolyCollection(polys[2:3], edgecolor='cornflowerblue',facecolor='cornflowerblue', closed=False)  plt.gca().add\_collection(lc)  lc = PolyCollection(polys[3:4], edgecolor='salmon',facecolor='salmon', closed=False)  plt.gca().add\_collection(lc)  lc = PolyCollection(polys[4:5],edgecolor='bisque', facecolor='bisque', closed=False)  plt.gca().add\_collection(lc)  lc = PolyCollection(polys[5:6],edgecolor='palevioletred', facecolor='palevioletred', closed=False)  plt.gca().add\_collection(lc)  lc = PolyCollection(polys[6:7],edgecolor='springgreen', facecolor='springgreen', closed=False)  plt.gca().add\_collection(lc)  lc = PolyCollection(polys[7:8], edgecolor='gold', facecolor='gold', closed=False)  plt.gca().add\_collection(lc)  lc = PolyCollection(polys[8:3000], edgecolor='bisque', facecolor='bisque', closed=False)  plt.gca().add\_collection(lc)  #============================================  x,y=m(long,lat)  #============================================  y\_offset=15  rotation=-30  me=160  size\_factor = 2.0  for i,j,g,c,e in zip(x,y,group,id,economic1992):  if g=="[Africa]":  size = size\_factor\*e/me  cs=m.scatter(i,j,s=size,marker='o',color='yellow')  plt.text(i,j+y\_offset,c,rotation=-rotation,fontsize=8)  if g=="[Asia]":  size = size\_factor\*e/me  cs=m.scatter(i,j,s=size,marker='o',color='sienna')  plt.text(i,j+y\_offset,c,rotation=-rotation,fontsize=8)  if g=="[Europe]":  size = size\_factor\*e/me  cs=m.scatter(i,j,s=size,marker='o',color='aqua')  plt.text(i,j+y\_offset,c,rotation=-rotation,fontsize=8)  if g=="[America]":  size = size\_factor\*e/me  cs=m.scatter(i,j,s=size,marker='o',color='mediumspringgreen')  plt.text(i,j+y\_offset,c,rotation=-rotation,fontsize=8)  #============================================  plt.title("McBryde-Thomas Flat Polar Quartic Projection")  plt.show() |

## 点地图：



## 网地图：



## 分析效果：

从图上清晰地看出：欧洲的GDP数据最为密集，其次是非洲，最不密集的是大洋洲和南极洲。