## 2.7.4 空间动力学——空间马尔可夫链

### 2.7.4.1 数据预处理

#### 1）[MCD12Q1\_v006](https://lpdaac.usgs.gov/products/mcd12q1v006/) 土地覆盖类型数据集[1]

由Terra 和 Aqua卫星搭载的中分辨率成像光谱仪（Moderate Resolution Imaging Spectroradiometer，MODIS），每隔1到2天观察整个地球表面，获取36个光谱带或一组波长的数据，这些数据可用于对发生在陆地、海洋和低层大气中的全球动态和过程的理解，MODIS在开发经过验证的全球交互式地球系统模型方面发挥着重要作用，这些模型能够准确地预测全球变化，足以帮助决策者做出有关保护地球环境的合理决定。基于MODIS的第6版土地覆盖类型（MCD12Q1）数据产品提供了按年度划分的全球土地覆盖类型（2001年至2020年）数据。该数据通过对MODIS Terra和Aqua反射率数据进行监督分类得出，经过额外的后处理，结合先前的知识和辅助信息，进一步完善了特定的类别。

MCD12Q1\_v006存储格式为HDF4，每一年的HDF4数据包括土地覆盖类型1-5、土地覆盖属性1-3、土地覆盖属性评估1-3、土地覆盖质量控制（QC）和土地水掩码的层，各层数据具体解释摘录如下：

| SDS Name | Description | Units | Data Type | Fill Value | No Data Value | Valid Range | Scale Factor |
| --- | --- | --- | --- | --- | --- | --- | --- |
| LC\_Type1 | Land Cover Type 1: Annual International Geosphere-Biosphere Programme (IGBP) classification土地覆盖类型1：国际地圈-生物圈计划（IGBP）年度分类 | Class | 8-bit unsigned integer | 255 | N/A | 1 to 17 | N/A |
| LC\_Type2 | Land Cover Type 2: Annual University of Maryland (UMD) classification 土地覆盖类型2：马里兰大学（UMD）的年度分类 | Class | 8-bit unsigned integer | 255 | N/A | 0 to 15 | N/A |
| LC\_Type3 | Land Cover Type 3: Annual Leaf Area Index (LAI) classification 土地覆盖类型3：年度叶面积指数（LAI）分类 | Class | 8-bit unsigned integer | 255 | N/A | 0 to 10 | N/A |
| LC\_Type4 | Land Cover Type 4: Annual BIOME-Biogeochemical Cycles (BGC) classification 土地覆盖类型4：年度生物群落-生物地球化学循环（BGC）分类 | Class | 8-bit unsigned integer | 255 | N/A | 0 to 8 | N/A |
| LC\_Type5 | Land Cover Type 5: Annual Plant Functional Types（PFT） classification 土地覆盖类型5：一年生植物功能类型分类 | Class | 8-bit unsigned integer | 255 | N/A | 0 to 11 | N/A |
| LC\_Prop1 | FAO-Land Cover Classification System 1 (LCCS1) land cover layer 粮农组织-土地覆盖分类系统1（LCCS1）土地覆盖层 | Class | 8-bit unsigned integer | 255 | N/A | 1 to 43 | N/A |
| LC\_Prop2 | FAO-LCCS2 land use layer 粮农组织-LCCS2土地利用层 | Class | 8-bit unsigned integer | 255 | N/A | 1 to 40 | N/A |
| LC\_Prop3 | FAO-LCCS3 surface hydrology layer 粮农组织-LCCS3地表水文层 | Class | 8-bit unsigned integer | 255 | N/A | 1 to 51 | N/A |
| LC\_Prop1\_Assessment | LCCS1 land cover layer confidence LCCS1土地覆盖层置信度 | Percent | 8-bit unsigned integer | 255 | N/A | 0 to 100 | N/A |
| LC\_Prop2\_Assessment | LCCS2 land use layer confidence LCCS2土地利用层的置信度 | Percent | 8-bit unsigned integer | 255 | N/A | 0 to 100 | N/A |
| LC\_Prop3\_Assessment | LCCS3 surface hydrology layer confidence LCCS3地表水文层的置信度 | Percent | 8-bit unsigned integer | 255 | N/A | 0 to 100 | N/A |
| QC | Product quality flags产品质量标志 | Quality Flag | 8-bit unsigned integer | 255 | N/A | 0 to 10 | N/A |
| LW | Binary land (class 2) / water (class 1) mask derived from MOD44W 来自MOD44W的二进制土地（类2）/水（类1）掩码 | Class | 8-bit unsigned integer | 255 | N/A | 1 to 2 | N/A |

不同分类内容和值及描述摘录如下：

* IGBP分类（LC\_Type1）

| Name | Value | Description |
| --- | --- | --- |
| Evergreen Needleleaf Forests（常绿针叶林） | 1 | Dominated by evergreen conifer trees (canopy >2m). Tree cover >60%. |
| Evergreen Broadleaf Forests（常绿阔叶林） | 2 | Dominated by evergreen broadleaf and palmate trees (canopy >2m). Tree cover >60%. |
| Deciduous Needleleaf Forests（落叶针叶林） | 3 | Dominated by deciduous needleleaf (larch) trees (canopy >2m). Tree cover >60%. |
| Deciduous Broadleaf Forests（落叶阔叶林） | 4 | Dominated by deciduous broadleaf trees (canopy >2m). Tree cover >60%. |
| Mixed Forests（混合林） | 5 | Dominated by neither deciduous nor evergreen (40-60% of each) tree type (canopy >2m). Tree cover >60%. |
| Closed Shrublands（封闭性灌木林地） | 6 | Dominated by woody perennials (1-2m height) >60% cover. |
| Open Shrublands（开放性灌木林地） | 7 | Dominated by woody perennials (1-2m height) 10-60% cover. |
| Woody Savannas（木质稀树草原） | 8 | Tree cover 30-60% (canopy >2m). |
| Savannas（稀树草原） | 9 | Tree cover 10-30% (canopy >2m). |
| Grasslands （草地） | 10 | Dominated by herbaceous annuals (<2m). |
| Permanent Wetlands（永久湿地） | 11 | Permanently inundated lands with 30-60% water cover and >10% vegetated cover. |
| Croplands（耕地） | 12 | At least 60% of area is cultivated cropland. |
| Urban and Built-up Lands（城市和建成区土地） | 13 | At least 30% impervious surface area including building materials, asphalt, and vehicles. |
| Cropland/Natural Vegetation Mosaics（耕地/自然植被镶嵌物） | 14 | Mosaics of small-scale cultivation 40-60% with natural tree, shrub, or herbaceous vegetation. |
| Permanent Snow and Ice（永久性冰雪） | 15 | At least 60% of area is covered by snow and ice for at least 10 months of the year. |
| Barren（荒地） | 16 | At least 60% of area is non-vegetated barren (sand, rock, soil) areas with less than 10% vegetation. |
| Water Bodies（水体） | 17 | At least 60% of area is covered by permanent water bodies. |
| Unclassiﬁed（未分类的） | 255 | Has not received a map label because of missing inputs. |

* UMD分类（LC Type2）

| Name | Value | Description |
| --- | --- | --- |
| Water bodies（水体） | 0 | At least 60% of area is covered by permanent water bodies. |
| Evergreen Needleleaf Forests（常绿针叶林） | 1 | Dominated by evergreen conifer trees (canopy >2m). Tree cover >60%. |
| Evergreen Broadleaf Forests（常绿阔叶林） | 2 | Dominated by evergreen broadleaf and palmate trees (canopy >2m). Tree cover >60%. |
| Deciduous Needleleaf Forests（落叶针叶林） | 3 | Dominated by deciduous needleleaf (larch) trees (canopy >2m). Tree cover >60%. |
| Deciduous Broadleaf Forests（落叶阔叶林） | 4 | Dominated by deciduous broadleaf trees (canopy >2m). Tree cover >60%. |
| Mixed Forests（混合林） | 5 | Dominated by neither deciduous nor evergreen (40-60% of each) tree type (canopy >2m). Tree cover >60%. |
| Closed Shrublands（封闭性灌木林地） | 6 | Dominated by woody perennials (1-2m height) >60% cover. |
| Open Shrublands（开放性灌木林地） | 7 | Dominated by woody perennials (1-2m height) 10-60% cover. |
| Woody Savannas（木质稀树草原） | 8 | Tree cover 30-60% (canopy >2m). |
| Savannas（稀树草原） | 9 | Tree cover 10-30% (canopy >2m). |
| Grasslands（草地） | 10 | Dominated by herbaceous annuals (<2m). |
| Permanent Wetlands（永久湿地） | 11 | Permanently inundated lands with 30-60% water cover and >10% vegetated cover. |
| Croplands（耕地） | 12 | At least 60% of area is cultivated cropland. |
| Urban and Built-up Lands（城市和建成区土地） | 13 | At least 30% impervious surface area including building materials, asphalt, and vehicles. |
| Cropland/Natural Vegetation Mosaics（耕地/自然植被镶嵌物） | 14 | Mosaics of small-scale cultivation 40-60% with natural tree, shrub, or herbaceous vegetation. |
| Non-Vegetated Lands（非植被地） | 15 | At least 60% of area is non-vegetated barren (sand, rock, soil)orpermanentsnowandice with less than 10% vegetation. |
| Unclassiﬁed（未分类的） | 255 | Has not received a map label because of missing inputs. |

* LAI分类（LC Type3）

| Name | Value | Description |
| --- | --- | --- |
| Water Bodies（水体） | 0 | At least 60% of area is covered by permanent water bodies. |
| Grasslands（草地） | 1 | Dominated by herbaceous annuals (<2m) including cereal croplands. |
| Shrublands（灌木丛地） | 2 | Shrub (1-2m) cover >10%. |
| Broadleaf Croplands（阔叶农田） | 3 | Dominated by herbaceous annuals (<2m) that are cultivated with broadleaf crops. |
| Savannas（稀树草原） | 4 | Between 10-60% tree cover (>2m). |
| Evergreen Broadleaf Forests（常绿阔叶林） | 5 | Dominated by evergreen broadleaf and palmate trees (>2m). Tree cover >60%. |
| Deciduous Broadleaf Forests（落叶阔叶林） | 6 | Dominated by deciduous broadleaf trees (>2m). Tree cover >60%. |
| Evergreen Needleleaf Forests（常绿针叶林） | 7 | Dominated by evergreen conifer trees (>2m). Tree cover >60%. |
| Deciduous Needleleaf Forests（落叶针叶林） | 8 | Dominated by deciduous needleleaf (larch) trees (>2m). Tree cover >60%. |
| Non-Vegetated Lands（非植被性土地） | 9 | At least 60% of area is non-vegetated barren (sand, rock, soil)or permanent snow and ice with less than 10% vegetation. |
| Urban and Built-up Lands（城市和建成区土地） | 10 | At least 30% impervious surface area including building materials, asphalt, and vehicles. |
| Unclassiﬁed（未分类的） | 255 | Has not received a map label because of missing inputs. |

* BGC分类（LC Type4）

| Name | Value | Description |
| --- | --- | --- |
| Water Bodies（水体） | 0 | At least 60% of area is covered by permanent water bodies. |
| Evergreen Needleleaf Vegetation（常绿针叶植被） | 1 | Dominated by evergreen conifer trees and shrubs (>1m). Woody vegetation cover >10%. |
| Evergreen Broadleaf Vegetation（常绿阔叶植被） | 2 | Dominated by evergreen broadleaf and palmate treesandshrubs(>1m). Woodyvegetationcover >10%. |
| Deciduous Needleleaf Vegetation（落叶针叶植被） | 3 | Dominated by deciduous needleleaf (larch) trees and shrubs (>1m). Woody vegetation cover >10%. |
| Deciduous Broadleaf Vegetation（落叶阔叶植被） | 4 | Dominated by deciduous broadleaf trees and shrubs (>1m). Woody vegetation cover >10%. |
| Annual Broadleaf Vegetation（一年生阔叶植被） | 5 | Dominated by herbaceous annuals (<2m). At least 60% cultivated broadleaf crops. |
| Annual Grass Vegetation（一年生草类植被） | 6 | Dominated by herbaceous annuals (<2m) including cereal croplands. |
| Non-Vegetated Lands（非植被地） | 7 | At least 60% of area is non-vegetated barren (sand,rock,soil)orpermanentsnow/icewithless than 10% vegetation. |
| Urban and Built-up Lands（城市和建成区土地） | 8 | At least 30% impervious surface area including building materials, asphalt, and vehicles. |
| Unclassiﬁed（未分类的） | 255 | Has not received a map label because of missing inputs. |

* PFT分类（LC Type5）

| Name | Value | Description |
| --- | --- | --- |
| Water Bodies（水体） | 0 | At least 60% of area is covered by permanent water bodies. |
| Evergreen Needleleaf Trees（常绿针叶树） | 1 | Dominated by evergreen conifer trees (>2m). Tree cover >10%. |
| Evergreen Broadleaf Trees（常绿阔叶树） | 2 | Dominated by evergreen broadleaf and palmate trees (>2m). Tree cover >10%. |
| Deciduous Needleleaf Trees（落叶针叶树） | 3 | Dominated by deciduous needleleaf (larch) trees (>2m). Tree cover >10%. |
| Deciduous Broadleaf Trees（落叶阔叶树） | 4 | Dominated by deciduous broadleaf trees (>2m). Tree cover >10%. |
| Shrub（灌木） | 5 | Shrub (1-2m) cover >10%. |
| Grass（草地） | 6 | Dominated by herbaceous annuals (<2m) that are not cultivated. |
| Cereal Croplands（谷类耕地） | 7 | Dominated by herbaceous annuals (<2m). At least 60% cultivated cereal crops. |
| Broadleaf Croplands（阔叶农田） | 8 | Dominated by herbaceous annuals (<2m). At least 60% cultivated broadleaf crops. |
| Urban and Built-up Lands（城市和建成区土地） | 9 | At least 30% impervious surface area including building materials, asphalt, and vehicles. |
| Permanent Snow and Ice（永久性的雪和冰） | 10 | At least 60% of area is covered by snow and ice for at least 10 months of the year. |
| Barren（荒地） | 11 | At least 60% of area is non-vegetated barren (sand, rock, soil) with less than 10% vegetation. |
| Unclassiﬁed（未分类的） | 255 | Has not received a map label because of missing inputs. |

* LCCS1分类（LC\_Prop1）

| Name | Value | Description |
| --- | --- | --- |
| Barren（贫瘠） | 1 | At least of area 60% is non-vegetated barren (sand,rock,soil)orpermanentsnow/icewithless than 10% vegetation. |
| Permanent Snow and Ice（永久的雪和冰） | 2 | At least of area 60% is covered by snow and ice for at least 10 months of the year. |
| Water Bodies（水体） | 3 | At least 60% of area is covered by permanent water bodies. |
| Evergreen Needleleaf Forests（常绿针叶林） | 11 | Dominated by evergreen conifer trees (>2m). Tree cover >60%. |
| Evergreen Broadleaf Forests（常绿阔叶林） | 12 | Dominated by evergreen broadleaf and palmate trees (>2m). Tree cover >60%. |
| Deciduous Needleleaf Forests（落叶针叶林） | 13 | Dominated by deciduous needleleaf (larch) trees (>2m). Tree cover >60%. |
| Deciduous Broadleaf Forests（落叶阔叶林） | 14 | Dominated by deciduous broadleaf trees (>2m). Tree cover >60%. |
| Mixed Broadleaf/Needleleaf Forests（阔叶/针叶混交林） | 15 | Co-dominated (40-60%) by broadleaf deciduous and evergreen needleleaf tree (>2m) types. Tree cover >60%. |
| Mixed Broadleaf Evergreen/Deciduous Forests（常绿/落叶混合阔叶林） | 16 | Co-dominated (40-60%) by broadleaf evergreen and deciduous tree (>2m) types. Tree cover >60%. |
| Open Forests（疏林） | 21 | Tree cover 30-60% (canopy >2m). |
| Sparse Forests（稀疏的森林） | 22 | Tree cover 10-30% (canopy >2m). |
| Dense Herbaceous（浓密的草本植物） | 31 | Dominated by herbaceous annuals(<2m) at least 60% cover. |
| Sparse Herbaceous（稀疏的草本植物） | 32 | Dominated by herbaceous annuals(<2m)10-60% cover. |
| Dense Shrublands（茂密的灌木林地） | 41 | Dominated by woody perennials (1-2m) >60% cover. |
| Shrubland/Grassland Mosaics（灌木丛/草原镶嵌物） | 42 | Dominated by woody perennials (1-2m) 10-60% cover with dense herbaceous annual understory. |
| Sparse Shrublands（稀疏的灌木丛） | 43 | Dominated by woody perennials (1-2m) 10-60% cover with minimal herbaceous understory. |
| Unclassiﬁed（未分类的） | 255 | Has not received a map label because of missing inputs. |

* LCCS2分类（LC\_Prop2）

| Name | Value | Description |
| --- | --- | --- |
| Barren（贫瘠） | 1 | At least 60% of area is non-vegetated barren (sand,rock,soil)orpermanentsnow/icewithless than 10% vegetation. |
| Permanent Snow and Ice（永久性的雪和冰） | 2 | At least 60% of area is covered by snow and ice for at least 10 months of the year. |
| Water Bodies（水体） | 3 | At least 60% of area is covered by permanent water bodies. |
| Urban and Built-up Lands（城市和建筑用地） | 9 | At least 30% of area is made up of impervious surfaces including building materials, asphalt, and vehicles. |
| Dense Forests（茂密的森林） | 10 | Tree cover >60% (canopy >2m). |
| Open Forests（开放的森林） | 20 | Tree cover 10-60% (canopy >2m). |
| Forest/Cropland Mosaics（森林/耕地镶嵌物） | 25 | Mosaics of small-scale cultivation 40-60% with >10% natural tree cover. |
| Natural Herbaceous（天然草本植物） | 30 | Dominated by herbaceous annuals (<2m). At least 10% cover. |
| Natural Herbaceous/Croplands Mosaics（天然草木/耕地镶嵌物） | 35 | Mosaics of small-scale cultivation 40-60% with natural shrub or herbaceous vegetation. |
| Herbaceous Croplands（草本农田） | 36 | Dominated by herbaceous annuals (<2m). At least 60% cover. Cultivated fraction >60%. |
| Shrublands（灌木地） | 40 | Shrub cover >60% (1-2m). |
| Unclassiﬁed（未分类的） | 255 | Has not received a map label because of missing inputs. |

* LCCS3（LC\_Prop3）

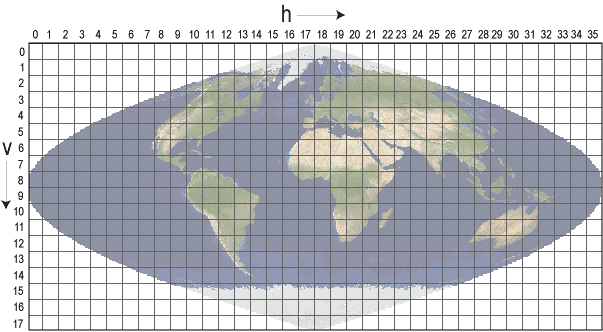
| Name | Value | Description |
| --- | --- | --- |
| Barren（贫瘠） | 1 | At least 60% of area is non-vegetated barren (sand,rock,soil) or permanent snow/ice withless than 10% vegetation. |
| Permanent Snow and Ice（永久的雪和冰） | 2 | At least 60% of area is covered by snow and ice for at least 10 months of the year. |
| Water Bodies （水体） | 3 | At least 60% of area is covered by permanent water bodies. |
| Dense Forests（茂密的森林） | 10 | Tree cover >60% (canopy >2m). |
| Open Forests（疏林） | 20 | Tree cover 10-60% (canopy >2m). |
| Woody Wetlands（木质湿地） | 27 | Shrub and tree cover >10% (>1m). Permanently or seasonally inundated |
| Grasslands（草地） | 30 | Dominated by herbaceous annuals (<2m) >10% cover. |
| Shrublands （灌木丛地） | 40 | Shrub cover >60% (1-2m). |
| Herbaceous Wetlands（草本湿地） | 50 | Dominated by herbaceous annuals (<2m) >10% cover. Permanently or seasonally inundated. |
| Tundra （苔原） | 51 | Tree cover <10%. Snow-covered for at least 8 months of the year. |
| Unclassiﬁed（未分类的） | 255 | Has not received a map label because of missing inputs. |

[**MODIS网格（ MODIS Grids）**](https://modis-land.gsfc.nasa.gov/MODLAND_grid.html)**①**

MODIS陆地产品以4种分辨率（250m、500m、1km和0.05degree）和3种投影（Sinusoidal, Lambert Azimuthal Equal-Area, and Geographic）制作。简单的地理纬度投影只用于最粗分辨率的网格，产生于0.05km（~5.5km）分辨率的网格被称为气候模型网格（Climate Modeling Grid，CMG）。为了使其它更高分辨率的MODIS陆地数据产品保持合理的文件大小，每个投影都被划分为一个平铺的网格。因此，土地产品被制作并分布在相邻不重叠的瓦片（tiles）中，这些瓦片大约为10度见方（在赤道）。大多数较高分辨率的MODIS陆地产品都是在正弦形瓦片网格中制作的，只有海冰产品是在极地Lambert Azimuthal Equal-Area瓦片网格中制作。

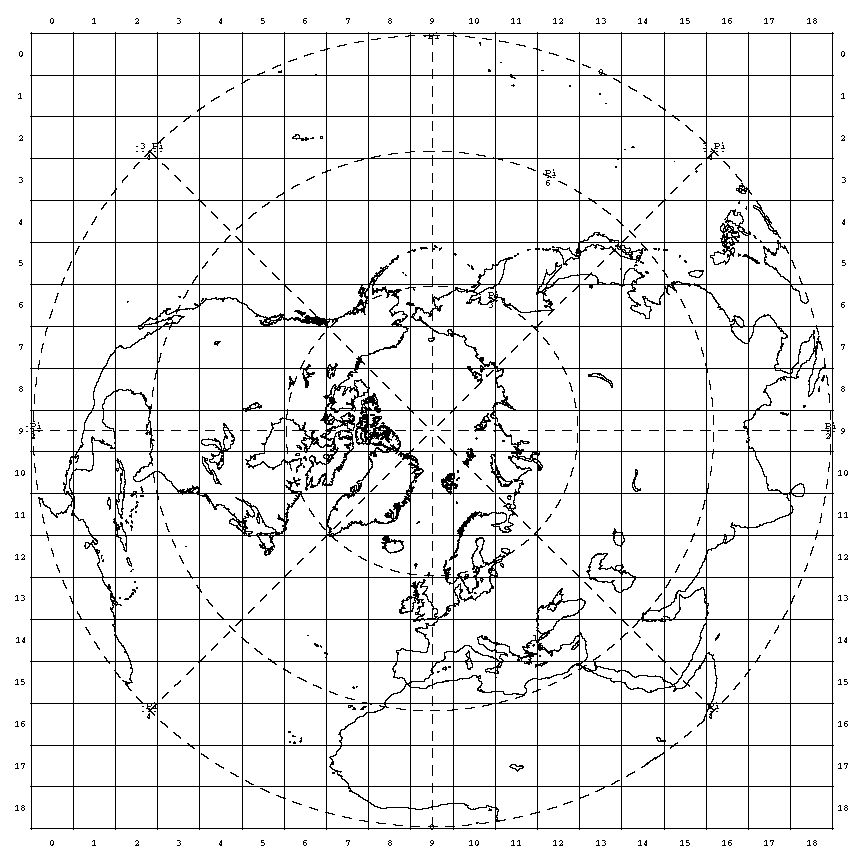
* Sinusoidal Tile Grid

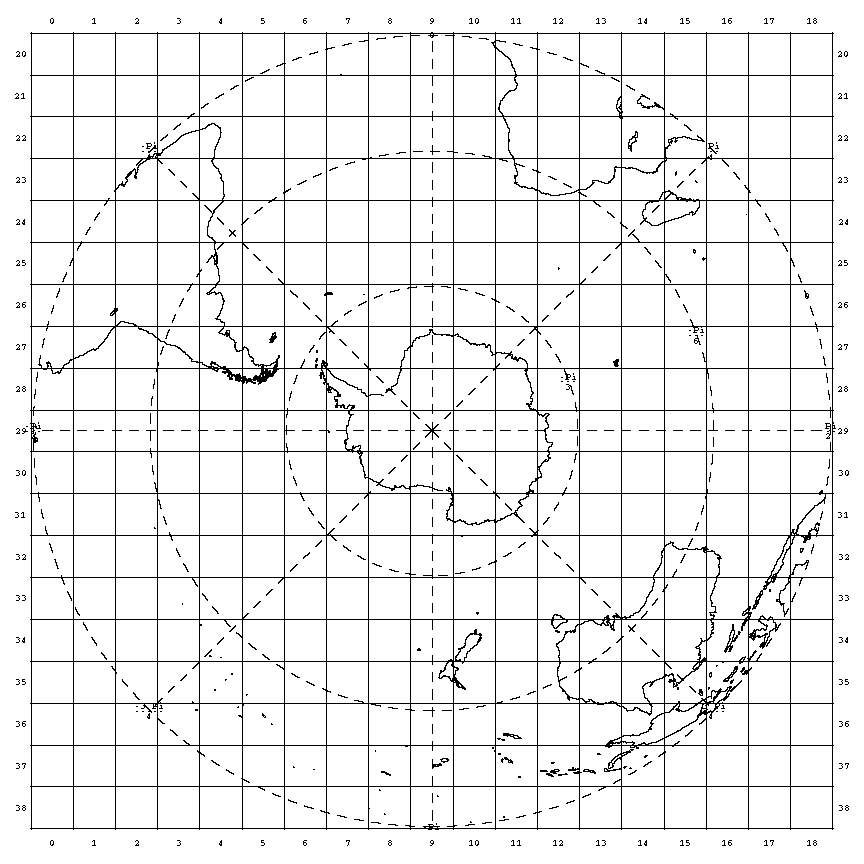
有460个非填充瓦片，瓦片在赤道上是10度乘10度。瓦片坐标系从左上角的（0,0）（水平瓦片编号，垂直瓦片编号）开始，向右（水平）和向下（垂直）前进。右下角的瓦片是（35,17）。



* Lambert Azimuthal Equal-Area Tile Grids (Polar)

网格中共有626块10度乘10度的非填充瓦片。一半的瓦片（313）在北极网格，另一半在南极网格。北极网格瓦片坐标系统从左上角的（0，0）（水平瓦片号，垂直瓦片号）开始，向右（水平）和向下（垂直）前进。右下角的瓦片是（18，18）。南极网格瓦片坐标系统从北极网格的终点开始，在（0，20），右下角的瓦片是（18，38）。极地网格是基于以每个极点为中心的Lambert Azimuthal Equal Area地图投影。这些网格与NSIDC EASI网格兼容。

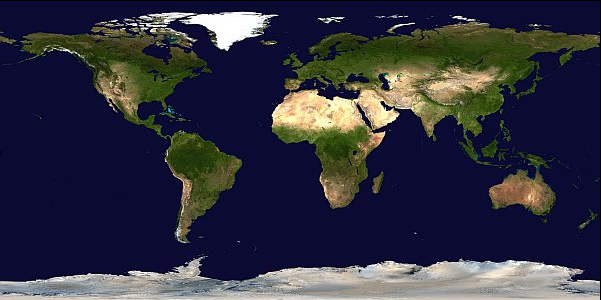




**地理投影（Geographical Projections）**

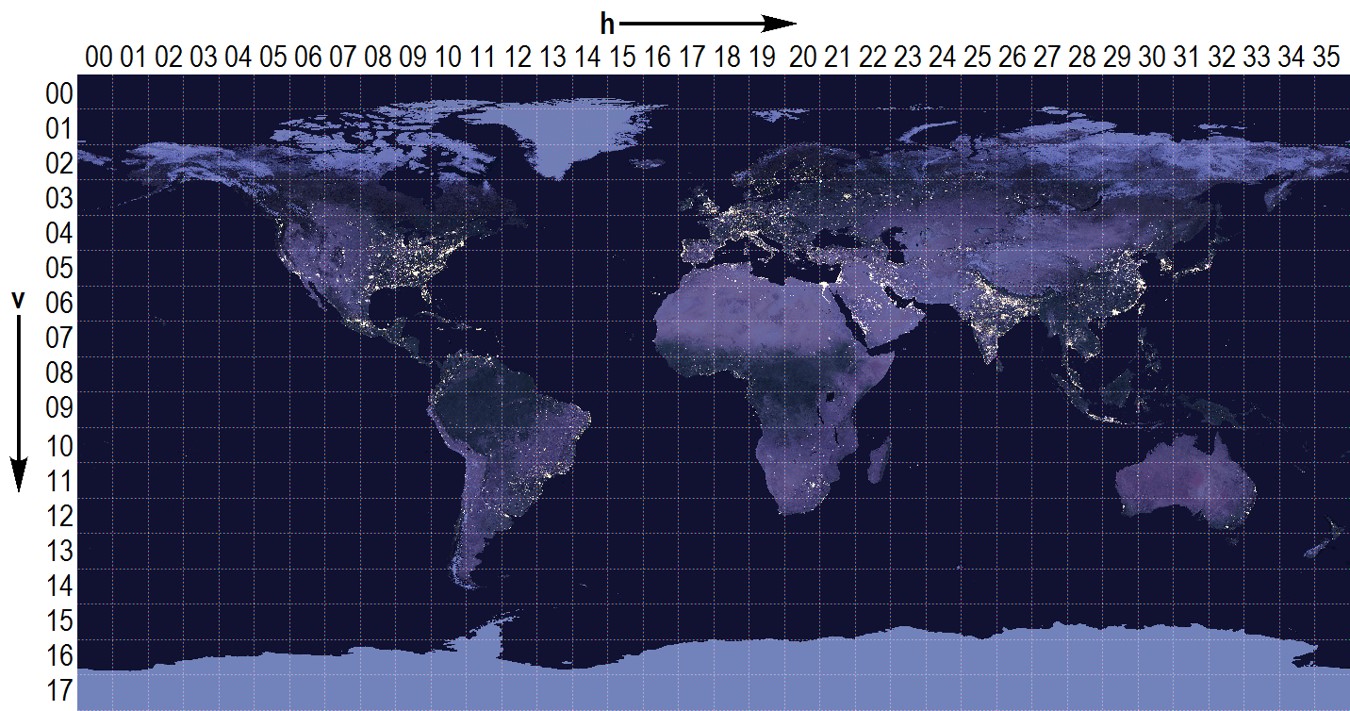
* Climate Modeling Grid (CMG)

气候模型网格（CMG）数据集以地理纬度投影提供全球覆盖，分辨率为0.05度。MODIS CMG图像的左上角像素的地理坐标为-180.00度经度，90.00度纬度。右下角像素的地理坐标是经度180.00度，纬度-90.00度。



* Linear Latitude/Longitude Grid

MODIS洪水图（MCDWD）产品使用地理线性经纬度投影。这种地理投影由460个不重叠的土地瓦片组成，尺寸约为10°X10°。瓦片坐标系统从左上角的（00,00）（水平瓦片编号，垂直瓦片编号）开始，向右（水平）和向下（垂直）前进。这种投影也被VIIRS DNB (VNP46)网格化产品所使用。



美国国家航空和航天局（National Aeronautics and Space Administration，NASA）的[MODIS Land](https://modis-land.gsfc.nasa.gov/MODLAND_grid.html)①网页提供了相应可下载的文本文件或在线说明，包括Sinusoidal Tile Grid的[Table of Tile Bounding Coordinates (10 deg tiles)](https://modis-land.gsfc.nasa.gov/pdf/sn_bound_10deg.txt)②，[Table of Tile G-ring Coordinates (10 deg tiles)](https://modis-land.gsfc.nasa.gov/pdf/sn_gring_10deg.txt)③和[General Cartographic Transformation Package (GCTP) projection parameters and other tile infomation](https://modis-land.gsfc.nasa.gov/GCTP.html)④；Lambert Azimuthal Equal-Area Tile Grids (Polar)的[Table of Tile Bounding Coordinates](https://modis-land.gsfc.nasa.gov/pdf/MODISPolarGridTiles.pdf)⑤;及Linear Latitude/Longitude Grid的[lat/long bounding (G-ring) coordinates](https://modis-land.gsfc.nasa.gov/lat_long_tiles.txt)⑥。

此次实验数据为[MCD12Q1\_v006](https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/)⑦，包括2001年至2020年的数据，每年的数据按照Sinusoidal Tile Grid方式切分为单独的HDF文件，例如2020年数据文件夹下的“MCD12Q1.A2020001.h21v03.006.2021360001124.hdf”数据文件为h21v03位置瓦片数据，即水平瓦片号为21，垂直瓦片号为03。Sinusoidal Tile Grid划分方式包括460个瓦片，分析区域仅为中国区域的数据，因此无需下载全部数据，但是需要根据下载的Table of Tile Bounding Coordinates (10 deg tiles)文件（默认下载文件名为“sn\_bound\_10deg.txt”），提取分析区域瓦片的水平瓦片号（ih）和垂直瓦片号（iv），定义函数modland\_grids()实现自动化批量提取工作。

参数定义和参数值的存储仍使用自定义的AttrDict类方法，方便常用参数值的管理和扩充，初步定义有存储投影信息的gi子属性，和存储有数据文件路径的data子属性。

from database import postSQL2gpd,gpd2postSQL  
from util\_misc import AttrDict  
\_\_C=AttrDict()   
args=\_\_C  
  
\_\_C.gi=AttrDict()  
\_\_C.gi.Chicago\_epsg=32616  
\_\_C.gi.epsg\_wgs84=4326  
  
\_\_C.data=AttrDict()  
\_\_C.data.sn\_bound\_10deg='./data/MCD12Q1v006/sn\_bound\_10deg.txt' # Table of Tile Bounding Coordinates (10 deg tiles)文件  
\_\_C.data.MCD12Q1v006\_fns=r'G:\data\MCD12Q1v006\data\all' # MCD12Q1v006数据文件所在文件夹  
\_\_C.data.CH\_provincial\_borders\_fn='./data/CN-sheng-A/CN-sheng-A.shp' # 2015年中国省级行政边界数据  
\_\_C.data.MCD12Q1v006\_preprocessed=r'G:/data/MCD12Q1v006/MCD12Q1v006\_CH.nc' # 放置MCD12Q1v006处理后数据  
\_\_C.data.MCD12Q1v006\_dt\_vals=r'G:/data/MCD12Q1v006/MCD12Q1v006\_dt\_vals.pickle' # 放置对位坐标的历年分类值  
\_\_C.data.MCD12Q1v006\_tif=r'D:\data\MCD12Q1v006\tif' # 放置MCD12Q1v006转换为栅格.tif格式文件  
\_\_C.data.MCD12Q1v006\_years\_vals\_array=r'D:\data\MCD12Q1v006\MCD12Q1v006\_years\_vals\_array.pickle' # 合并MCD12Q1v006粮农组织-土地覆盖分类系统1（LCCS1）土地覆盖层历年数据为一个数组

import pandas as pd  
pd.options.mode.chained\_assignment=None # default='warn'  
  
def modland\_grids(fp,start\_idx=0,boundary\_coordi=None):   
 '''  
 给定经纬度范围，提取Sinusoidal Tile Grid方式下，提取MCD12Q1\_v006数据集对应官方文件Table of Tile Bounding Coordinates (10 deg tiles)满足要求的行信息，包括水平瓦片号（ih）和垂直瓦片号（iv）  
  
 Parameters  
 ----------  
 fp : string  
 Table of Tile Bounding Coordinates (10 deg tiles)文件，默认名为sn\_bound\_10deg.txt.  
 start\_idx : int, optional  
 开始行，忽略抬头信息. The default is 0.  
 boundary\_coordi : list(float), optional  
 提取边界经纬度信息，格式为 [lon\_min,lon\_max,lat\_min,lat\_max]. The default is None.  
  
 Returns  
 -------  
 DataFrame  
 位于给定范围内的数据行信息，包括水平瓦片号（ih）和垂直瓦片号（iv.  
   
 '''   
 import pandas as pd  
   
 with open (fp,'r') as f:  
 lines=f.readlines()[start\_idx:-2]  
 lines\_df=pd.DataFrame(columns=lines[0].split(),data=[row.split() for row in lines[1:]])  
 lines\_df=lines\_df.apply(pd.to\_numeric,errors='ignore')  
   
 def within\_func(row):  
 lon\_sum=(row.lon\_min>=lon\_min and row.lon\_min<=lon\_max)+(row.lon\_max>=lon\_min and row.lon\_max<=lon\_max)   
 lat\_sum=(row.lat\_min>=lat\_min and row.lat\_min<=lat\_max)+(row.lat\_max>=lat\_min and row.lat\_max<=lat\_max)  
 if lon\_sum>0 and lat\_sum>0:  
 return 1  
 else:   
 return 0   
   
 if boundary\_coordi:  
 lon\_min,lon\_max,lat\_min,lat\_max=boundary\_coordi  
 lines\_df['mask']=lines\_df.apply(within\_func,axis=1)   
 extracted\_rows=lines\_df[lines\_df['mask']==1]  
 extracted\_rows['identifier']=extracted\_rows.apply(lambda row:f"h{int(row.ih)}v{int(row.iv):02}",axis=1)  
 return extracted\_rows  
 else:  
 return lines\_df   
   
modland\_ivih\_masked=modland\_grids(args.data.sn\_bound\_10deg,start\_idx=6,boundary\_coordi=[68,139,15,57])   
print(modland\_ivih\_masked.shape)  
modland\_ivih\_masked.sort\_values(by=['ih','iv'])

(38, 8)

|  | **iv** | **ih** | **lon\_min** | **lon\_max** | **lat\_min** | **lat\_max** | **mask** | **identifier** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **129** | 3 | 21 | 46.6717 | 80.0167 | 50.0 | 60.0 | 1 | h21v03 |
| **130** | 3 | 22 | 62.2290 | 100.0167 | 50.0 | 60.0 | 1 | h22v03 |
| **166** | 4 | 22 | 52.2163 | 77.7992 | 40.0 | 50.0 | 1 | h22v04 |
| **131** | 3 | 23 | 77.7862 | 120.0167 | 50.0 | 60.0 | 1 | h23v03 |
| **167** | 4 | 23 | 65.2704 | 93.3564 | 40.0 | 50.0 | 1 | h23v04 |
| **203** | 5 | 23 | 57.7350 | 78.3353 | 30.0 | 40.0 | 1 | h23v05 |
| **239** | 6 | 23 | 53.2089 | 69.2917 | 20.0 | 30.0 | 1 | h23v06 |
| **132** | 3 | 24 | 93.3434 | 140.0167 | 50.0 | 60.0 | 1 | h24v03 |
| **168** | 4 | 24 | 78.3244 | 108.9136 | 40.0 | 50.0 | 1 | h24v04 |
| **204** | 5 | 24 | 69.2820 | 91.3894 | 30.0 | 40.0 | 1 | h24v05 |
| **240** | 6 | 24 | 63.8507 | 80.8387 | 20.0 | 30.0 | 1 | h24v06 |
| **276** | 7 | 24 | 60.9256 | 74.5013 | 10.0 | 20.0 | 1 | h24v07 |
| **133** | 3 | 25 | 108.9007 | 160.0167 | 50.0 | 60.0 | 1 | h25v03 |
| **169** | 4 | 25 | 91.3785 | 124.4709 | 40.0 | 50.0 | 1 | h25v04 |
| **205** | 5 | 25 | 80.8290 | 104.4435 | 30.0 | 40.0 | 1 | h25v05 |
| **241** | 6 | 25 | 74.4924 | 92.3857 | 20.0 | 30.0 | 1 | h25v06 |
| **277** | 7 | 25 | 71.0799 | 85.1431 | 10.0 | 20.0 | 1 | h25v07 |
| **134** | 3 | 26 | 124.4579 | 180.0000 | 50.0 | 60.0 | 1 | h26v03 |
| **170** | 4 | 26 | 104.4326 | 140.0281 | 40.0 | 50.0 | 1 | h26v04 |
| **206** | 5 | 26 | 92.3760 | 117.4975 | 30.0 | 40.0 | 1 | h26v05 |
| **242** | 6 | 26 | 85.1342 | 103.9327 | 20.0 | 30.0 | 1 | h26v06 |
| **278** | 7 | 26 | 81.2341 | 95.7849 | 10.0 | 20.0 | 1 | h26v07 |
| **171** | 4 | 27 | 117.4867 | 155.5853 | 40.0 | 50.0 | 1 | h27v04 |
| **207** | 5 | 27 | 103.9230 | 130.5516 | 30.0 | 40.0 | 1 | h27v05 |
| **243** | 6 | 27 | 95.7760 | 115.4797 | 20.0 | 30.0 | 1 | h27v06 |
| **279** | 7 | 27 | 91.3884 | 106.4266 | 10.0 | 20.0 | 1 | h27v07 |
| **172** | 4 | 28 | 130.5407 | 171.1426 | 40.0 | 50.0 | 1 | h28v04 |
| **208** | 5 | 28 | 115.4701 | 143.6057 | 30.0 | 40.0 | 1 | h28v05 |
| **244** | 6 | 28 | 106.4178 | 127.0267 | 20.0 | 30.0 | 1 | h28v06 |
| **280** | 7 | 28 | 101.5427 | 117.0684 | 10.0 | 20.0 | 1 | h28v07 |
| **209** | 5 | 29 | 127.0171 | 156.6598 | 30.0 | 40.0 | 1 | h29v05 |
| **245** | 6 | 29 | 117.0596 | 138.5737 | 20.0 | 30.0 | 1 | h29v06 |
| **281** | 7 | 29 | 111.6969 | 127.7102 | 10.0 | 20.0 | 1 | h29v07 |
| **210** | 5 | 30 | 138.5641 | 169.7138 | 30.0 | 40.0 | 1 | h30v05 |
| **246** | 6 | 30 | 127.7013 | 150.1207 | 20.0 | 30.0 | 1 | h30v06 |
| **282** | 7 | 30 | 121.8512 | 138.3520 | 10.0 | 20.0 | 1 | h30v07 |
| **247** | 6 | 31 | 138.3431 | 161.6677 | 20.0 | 30.0 | 1 | h31v06 |
| **283** | 7 | 31 | 132.0055 | 148.9938 | 10.0 | 20.0 | 1 | h31v07 |

modland\_grids()函数计算返回的DataFrame增加了一个新列identifier，该列对应下载文件名例如“MCD12Q1.A2020001.h21v03.006.2021360001124.hdf”中的h21v03，指明下载瓦片区域的水平瓦片号（ih）和垂直瓦片号（iv），用于网页数据批量检索提取的标注信息。

identifier=modland\_ivih\_masked.identifier.to\_list()  
print(identifier)

['h21v03', 'h22v03', 'h23v03', 'h24v03', 'h25v03', 'h26v03', 'h22v04', 'h23v04', 'h24v04', 'h25v04', 'h26v04', 'h27v04', 'h28v04', 'h23v05', 'h24v05', 'h25v05', 'h26v05', 'h27v05', 'h28v05', 'h29v05', 'h30v05', 'h23v06', 'h24v06', 'h25v06', 'h26v06', 'h27v06', 'h28v06', 'h29v06', 'h30v06', 'h31v06', 'h24v07', 'h25v07', 'h26v07', 'h27v07', 'h28v07', 'h29v07', 'h30v07', 'h31v07']

**网络抓取（Web crawling）**

下述代码迁移于[Web crawling with Python](https://www.scrapingbee.com/blog/crawling-python/)⑧中简单的网络抓取代码部分，并做了简单的调整，增加了是否搜索子页的iter初始化参数，如果为False则仅抓取指定页面可下载文件的链接。同时增加了visited\_urls\_dict变量，以字典形式存储页面和对应页面可下载文件链接。要在Python中建立一个简单的网络爬虫，至少需要一个库来从URL中下载HTML，及一个HTML解析库来提取链接。Python提供了用于发出HTTP请求的标准库urllib和用于解析HTML的html.parser。用于请求和HTML解析的标准Python库对开发者不是很友好，可使用其它其他流行的库，例如request和Beautiful Soup等，提供更好的开发体验。下述定义的Crawler类，其中的辅助方法是包含使用requests库的get().text方法定义的download\_url()函数，包含使用BeautifulSoup库定义的get\_linked\_url()和add\_url\_to\_visit()函数过滤RUL。要访问的URL和已访问的URL被存储在两个独立的列表中，visited\_urls和visited\_urls。

import logging  
from urllib.parse import urljoin  
import requests  
from bs4 import BeautifulSoup  
  
logging.basicConfig(  
 format='%(asctime)s %(levelname)s:%(message)s',  
 level=logging.INFO)  
  
class Crawler:  
 '''  
 简单的网络抓取工具，调整迁移于：Web crawling with Python，https://www.scrapingbee.com/blog/crawling-python/  
 指定网页，抓取可以下载的文件链接。  
 '''  
 def \_\_init\_\_(self, urls=[],iter=True):  
 self.visited\_urls = []  
 self.urls\_to\_visit = urls  
 self.iter=iter  
 self.visited\_urls\_dict={}  
  
 def download\_url(self, url):  
 return requests.get(url).text  
  
 def get\_linked\_urls(self, url, html):  
 soup = BeautifulSoup(html, 'html.parser')  
 for link in soup.find\_all('a'):  
 path = link.get('href')  
 if path and path.startswith('/'):  
 path = urljoin(url, path)  
 yield path  
  
 def add\_url\_to\_visit(self, url):  
 if url not in self.visited\_urls and url not in self.urls\_to\_visit:  
 self.urls\_to\_visit.append(url)  
  
 def crawl(self, url\_):  
 self.visited\_urls\_dict[url\_]=[]  
 html = self.download\_url(url\_)  
 for url in self.get\_linked\_urls(url\_, html):  
 if self.iter:  
 self.add\_url\_to\_visit(url)  
 else:  
 self.visited\_urls\_dict[url\_].append(url)  
  
 def run(self):  
 while self.urls\_to\_visit:  
 url = self.urls\_to\_visit.pop(0)  
 logging.info(f'Crawling: {url}')   
 try:  
 self.crawl(url)  
 except Exception:  
 logging.exception(f'Failed to crawl: {url}')  
 finally:  
 self.visited\_urls.append(url)   
   
if \_\_name\_\_=="\_\_main\_\_":  
 url\_root\_lst=[f'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/{i}.01.01/' for i in range(2001,2021,1)]   
 wc=Crawler(urls=url\_root\_lst,iter=False)  
 wc.run()

抓取可获取的网页文件链接，存储在了visited\_urls\_dict字典中，为避免重复抓取，将该数据存储到本地空间，方便调用。

import pickle  
MCD12Q1\_crawl\_url\_fp="./data/MCD12Q1\_crawl\_urls.pickle"  
with open(MCD12Q1\_crawl\_url\_fp,'wb') as f:  
 pickle.dump(wc.visited\_urls\_dict,f)

with open(MCD12Q1\_crawl\_url\_fp,'rb') as f:  
 MCD12Q1\_crawl\_urls=pickle.load(f)

print(f"keys: {MCD12Q1\_crawl\_urls.keys()}")  
print("-"\*50)  
print(f"values: {MCD12Q1\_crawl\_urls['https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2001.01.01/'][:10]}")

keys: dict\_keys(['https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2001.01.01/', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2002.01.01/', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2003.01.01/', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2004.01.01/', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2005.01.01/', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2006.01.01/', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2007.01.01/', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2008.01.01/', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2009.01.01/', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2010.01.01/', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2011.01.01/', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2012.01.01/', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2013.01.01/', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2014.01.01/', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2015.01.01/', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2016.01.01/', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2017.01.01/', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2018.01.01/', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2019.01.01/', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2020.01.01/'])  
--------------------------------------------------  
values: ['https://urs.earthdata.nasa.gov/users/new', 'https://lpdaac.usgs.gov/', '?C=N;O=D', '?C=M;O=A', '?C=S;O=A', '?C=D;O=A', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/', 'MCD12Q1.A2001001.h00v08.006.2018142182903.hdf', 'MCD12Q1.A2001001.h00v08.006.2018142182903.hdf.xml', 'MCD12Q1.A2001001.h00v09.006.2018142182901.hdf']

通过modland\_grids()函数获取了分析区域的瓦片编号，通过Crawler类抓取了数据文件下载链接，定义urls\_extraction()函数，根据瓦片编号提取对应数据文件下载链接，并用NumPy库的savetxt()方法存储提取后的列表数据，方便直接用于浏览器扩展（Extensions）应用，例如Chrome浏览器的Simple mass downloader工具读取数据链接文件批量下载等。

import numpy as np  
  
def urls\_extraction(urls\_dict,conditional\_chars):  
 '''  
 根据modland\_grids()函数获取的瓦片编号，提取对应的Crawler类抓取的数据文件下载链接  
  
 Parameters  
 ----------  
 urls\_dict : list(string)  
 Crawler类抓取的数据文件下载链接.  
 conditional\_chars : list(string)  
 'hXvX'格式标识，例如'h21v03'，为水平瓦片号（ih）和垂直瓦片号（iv）.  
  
 Returns  
 -------  
 urls\_extraction\_dict : dict(string:string)  
 满足要求的数据文件下载链接，键为下载的文件夹，值为该文件夹下满足要求的数据文件下载链接.  
 urls\_extraction\_lst : list(string)  
 所有满足要求的数据文件下载链接列表.  
  
 '''   
 from urllib.parse import urljoin  
   
 urls\_extraction\_dict={}  
 urls\_extraction\_lst=[]  
 for k,v in urls\_dict.items():  
 urls\_extraction\_dict[k]=[]  
 for url in v:  
 for i in identifier:  
 if i in url:  
 urls\_extraction\_dict[k].append(urljoin(k,url))  
 urls\_extraction\_lst.append(urljoin(k,url))  
 return urls\_extraction\_dict,urls\_extraction\_lst   
   
urls\_extraction\_dict,urls\_extraction\_lst=urls\_extraction(MCD12Q1\_crawl\_urls,identifier)   
np.savetxt('./data/urls\_extraction\_lst.txt', urls\_extraction\_lst, delimiter="\n", fmt="%s")  
print(urls\_extraction\_lst[::200])  
print("-"\*50)  
print(f"符合要求的数据文件总数量: {len(urls\_extraction\_lst)}")

['https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2001.01.01/MCD12Q1.A2001001.h21v03.006.2018142213330.hdf', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2003.01.01/MCD12Q1.A2003001.h27v06.006.2018144212435.hdf', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2006.01.01/MCD12Q1.A2006001.h24v06.006.2018145212751.hdf', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2008.01.01/BROWSE.MCD12Q1.A2008001.h29v05.006.2018054191051.1.jpg', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2010.01.01/BROWSE.MCD12Q1.A2010001.h22v04.006.2018054201414.1.jpg', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2011.01.01/MCD12Q1.A2011001.h27v07.006.2018146004501.hdf', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2013.01.01/MCD12Q1.A2013001.h24v07.006.2018146014649.hdf', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2015.01.01/BROWSE.MCD12Q1.A2015001.h29v07.006.2018055065547.1.jpg', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2017.01.01/BROWSE.MCD12Q1.A2017001.h23v04.006.2019196135638.1.jpg', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2018.01.01/MCD12Q1.A2018001.h28v04.006.2019200010243.hdf', 'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/2020.01.01/MCD12Q1.A2020001.h25v03.006.2021360020156.hdf']  
--------------------------------------------------  
符合要求的数据文件总数量: 2052

获取分析区域MCD12Q1\_v006土地覆盖类型数据集数据文件下载链接后，可以用python继续编写下载程序，也可以直接使用浏览器扩展工具中的下载程序，这里使用了Chrome浏览器的Simple mass downloader工具加载保存的urls\_extraction\_lst.txt文件实现批量下载，下载的数据均置于了同一文件夹下。

import glob  
import os  
MCD12Q1v006\_2020\_fps=glob.glob(os.path.join(args.data.MCD12Q1v006\_fns,"\*.hdf"))  
MCD12Q1v006\_2020\_fps[:5]

['G:\\data\\MCD12Q1v006\\data\\all\\MCD12Q1.A2003001.h29v05.006.2018144213302.hdf',  
 'G:\\data\\MCD12Q1v006\\data\\all\\MCD12Q1.A2003001.h29v06.006.2018144213347.hdf',  
 'G:\\data\\MCD12Q1v006\\data\\all\\MCD12Q1.A2003001.h29v07.006.2018144213421.hdf',  
 'G:\\data\\MCD12Q1v006\\data\\all\\MCD12Q1.A2003001.h30v05.006.2018145033504.hdf',  
 'G:\\data\\MCD12Q1v006\\data\\all\\MCD12Q1.A2003001.h30v07.006.2018144214353.hdf']

下载的数据文件均位于同一文件夹下，需要按照年份分组提取。根据文件路径名可以观察到A2003001部分标识有数据年份，因此定义data\_grouping\_basedon\_formatting\_pattern()函数按给定的正则表达式提取标识字符，并以标识字符为键，以对应的路径名为值，以字典形式存储分组后数据。

def data\_grouping\_basedon\_formatting\_pattern(str\_data\_list,pattern):  
 '''  
 按给定的正则表达式分组数据列表  
  
 Parameters  
 ----------  
 str\_data\_list : list(string)  
 字符串列表.  
 pattern : string  
 正则表达式.  
  
 Returns  
 -------  
 data\_grouping : dict(string:string)  
 分组后字典.  
  
 '''   
 import re  
 data\_grouping={}  
 tabu=[]  
 for text in str\_data\_list:  
 string\_match=re.findall(pattern,text)[0]  
 if string\_match not in tabu:  
 tabu.append(string\_match)  
 data\_grouping[string\_match]=[text]  
 else:  
 data\_grouping[string\_match].append(text)  
 return data\_grouping  
  
pattern="(?x)MCD12Q1\.(.\*?)\."   
MCD12Q1v006\_group=data\_grouping\_basedon\_formatting\_pattern(MCD12Q1v006\_2020\_fps,pattern)  
print(MCD12Q1v006\_group.keys())

dict\_keys(['A2003001', 'A2004001', 'A2005001', 'A2006001', 'A2001001', 'A2007001', 'A2002001', 'A2008001', 'A2011001', 'A2009001', 'A2012001', 'A2013001', 'A2010001', 'A2014001', 'A2017001', 'A2015001', 'A2018001', 'A2019001', 'A2016001', 'A2020001'])

**颜色值**

打印分类地图时，需要配置类别色彩，matplotlib库提供了[plot\_colortable()](https://matplotlib.org/3.1.1/gallery/color/named_colors.html)函数⑨，可以直接迁移运行查看该库支持的不同色系颜色名称。

from matplotlib.patches import Rectangle  
import matplotlib.pyplot as plt  
import matplotlib.colors as mcolors  
  
def plot\_colortable(colors, sort\_colors=True, emptycols=0):  
 '''  
 绘制一个matplotlib支持命名颜色的列表，迁移于matplotlib库“List of named colors”，https://matplotlib.org/3.1.1/gallery/color/named\_colors.html  
  
 Parameters  
 ----------  
 colors : dict  
 由mcolors模块提取，例如mcolors.BASE\_COLORS，mcolors.TABLEAU\_COLORS，mcolors.CSS4\_COLORS等.  
 sort\_colors : bool, optional  
 是否排序颜色. The default is True.  
 emptycols : int, optional  
 调整列数量，避免列中出现空值. The default is 0.  
  
 Returns  
 -------  
 fig : matplotlib.figure.Figure  
 返回图表.  
  
 '''   
 cell\_width = 212  
 cell\_height = 22  
 swatch\_width = 48  
 margin = 12  
  
 # Sort colors by hue, saturation, value and name.  
 if sort\_colors is True:  
 by\_hsv = sorted((tuple(mcolors.rgb\_to\_hsv(mcolors.to\_rgb(color))),  
 name)  
 for name, color in colors.items())  
 names = [name for hsv, name in by\_hsv]  
 else:  
 names = list(colors)  
  
 n = len(names)  
 ncols = 4 - emptycols  
 nrows = n // ncols + int(n % ncols > 0)  
  
 width = cell\_width \* 4 + 2 \* margin  
 height = cell\_height \* nrows + 2 \* margin  
 dpi = 72  
  
 fig, ax = plt.subplots(figsize=(width / dpi, height / dpi), dpi=dpi)  
 fig.subplots\_adjust(margin/width, margin/height,  
 (width-margin)/width, (height-margin)/height)  
 ax.set\_xlim(0, cell\_width \* 4)  
 ax.set\_ylim(cell\_height \* (nrows-0.5), -cell\_height/2.)  
 ax.yaxis.set\_visible(False)  
 ax.xaxis.set\_visible(False)  
 ax.set\_axis\_off()  
  
 for i, name in enumerate(names):  
 row = i % nrows  
 col = i // nrows  
 y = row \* cell\_height  
  
 swatch\_start\_x = cell\_width \* col  
 text\_pos\_x = cell\_width \* col + swatch\_width + 7  
  
 ax.text(text\_pos\_x, y, name, fontsize=14,  
 horizontalalignment='left',  
 verticalalignment='center')  
  
 ax.add\_patch(  
 Rectangle(xy=(swatch\_start\_x, y-9), width=swatch\_width,  
 height=18, facecolor=colors[name], edgecolor='0.7')  
 )  
  
 return fig

plot\_colortable(mcolors.BASE\_COLORS, sort\_colors=False, emptycols=1)  
plot\_colortable(mcolors.TABLEAU\_COLORS, sort\_colors=True, emptycols=2)  
plot\_colortable(mcolors.CSS4\_COLORS)  
plt.show()

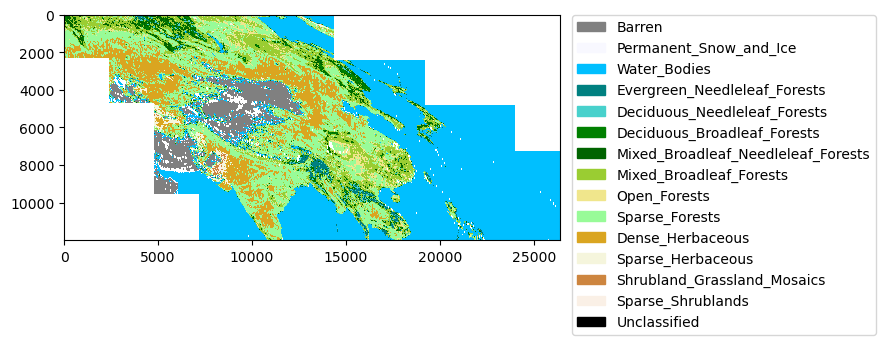


MCD12Q1\_v006 土地覆盖类型数据集下载完成后，需要通过打印地图的方式查看下载的数据瓦片是否完全并落于分析区域内。定义cmap\_patch\_build()函数为分类地图打印配置分类颜色和图例参数。在组织分类信息输入条件时，以字典形式给出如labels\_cmap\_dict，以分类值为键，以分类名和颜色值列表为值。定义MCD12Q1v006\_merge\_plot()函数打印地图，栅格数据处理主要使用的库为[rioxarray](https://github.com/corteva/rioxarray)⑩，MCD12Q1\_v006的HDF格式数据包含有多个层，打印时仅打印一层数据信息，因此需要指定层名称参数layer，下述打印的层名为LC\_Prop1，即粮农组织-土地覆盖分类系统1（LCCS1）土地覆盖层。

def cmap\_patch\_build(labels\_cmap\_dict):  
 '''  
 生成matplotlib.colors.ListedColormap，matplotlib.colors.BoundaryNorm和matplotlib.patches.Patch，用于分类数据地图打印并显示图例  
  
 Parameters  
 ----------  
 labels\_cmap\_dict : dict  
 字典格式为{分类值:[分类名，颜色则]}，例如{1:["Barren",'gray'],2:["Permanent\_Snow\_and\_Ice",'ghostwhite']}.  
  
 Returns  
 -------  
 cmap : matplotlib.colors.ListedColormap  
 Colormap object generated from a list of colors. it can also be used to generate special colormaps for ordinary mapping.  
 norm : matplotlib.colors.BoundaryNorm  
 Generate a colormap index based on discrete intervals.  
 patches : matplotlib.patches.Patch  
 A patch is a 2D artist with a face color and an edge color.  
  
 '''   
 from matplotlib.colors import from\_levels\_and\_colors  
 from matplotlib import colors  
 import matplotlib.patches as mpatches  
   
 labels={k:v[0] for k,v in labels\_cmap\_dict.items()}  
 cmap\_lst=list(map(list,zip(\*[[k,v[1]] for k,v in labels\_cmap\_dict.items()])))  
  
 cmap\_dict={k[0]:colors.to\_rgb(k[1]) for k in zip(\*cmap\_lst)}  
 cmap, norm=from\_levels\_and\_colors(cmap\_lst[0],cmap\_lst[1][:-1])   
 patches=[mpatches.Patch(color=cmap\_dict[i],label=labels[i]) for i in cmap\_dict]  
   
 return cmap,norm,patches  
  
labels\_cmap\_dict={1:["Barren",'gray'],  
 2:["Permanent\_Snow\_and\_Ice",'ghostwhite'],  
 3:["Water\_Bodies",'deepskyblue'],  
 11:["Evergreen\_Needleleaf\_Forests",'teal'],  
 13:["Deciduous\_Needleleaf\_Forests",'mediumturquoise'],  
 14:["Deciduous\_Broadleaf\_Forests",'green'],  
 15:["Mixed\_Broadleaf\_Needleleaf\_Forests",'darkgreen'],  
 16:["Mixed\_Broadleaf\_Forests",'yellowgreen'],  
 21:["Open\_Forests",'khaki'],  
 22:["Sparse\_Forests",'palegreen'],  
 31:["Dense\_Herbaceous",'goldenrod'],  
 32:["Sparse\_Herbaceous",'beige'],  
 42:["Shrubland\_Grassland\_Mosaics",'peru'],  
 43:["Sparse\_Shrublands",'linen'],  
 255:["Unclassified",'black']}  
cmap,norm,patches=cmap\_patch\_build(labels\_cmap\_dict)  
cmap



def MCD12Q1v006\_merge\_plot(fns,layer,cmap,norm,patches,\*\*args):  
 '''  
 指定数据文件夹和层名称，打印MCD12Q1v006分类数据地图  
  
 Parameters  
 ----------  
 fns : list(string)  
 数据文件路径列表.  
 layer : string  
 MCD12Q1v006分类层.  
 cmap : matplotlib.colors.ListedColormap  
 Colormap object generated from a list of colors. it can also be used to generate special colormaps for ordinary mapping.  
 norm : matplotlib.colors.BoundaryNorm  
 Generate a colormap index based on discrete intervals.  
 patches : matplotlib.patches.Patch  
 A patch is a 2D artist with a face color and an edge color.  
 \*\*args : 关键字参数  
 包括： params=dict(box\_to\_anchor=(1.9, 1.02),  
 ncol=1,  
 fancybox=True).  
  
 Returns  
 -------  
 merged : xarray.core.dataarray.DataArray  
 合并后的MCD12Q1v006分类数据.  
  
 '''   
 from rioxarray.merge import merge\_arrays  
 import rioxarray as rxr  
 import matplotlib.pyplot as plt  
   
 params=dict(box\_to\_anchor=(1.9, 1.02),  
 ncol=1,  
 fancybox=True)  
 params.update(args)  
   
 arrays=[rxr.open\_rasterio(i,masked=True)[layer] for i in fns]  
 merged=merge\_arrays(arrays)  
 plt.imshow(merged[0], cmap=cmap, norm=norm)  
 plt.legend(handles=patches, bbox\_to\_anchor=params['box\_to\_anchor'],ncol=params['ncol'], fancybox=params['fancybox'],)  
   
 return merged  
   
LCCS1\_merged=MCD12Q1v006\_merge\_plot(MCD12Q1v006\_group['A2002001'],'LC\_Prop1',cmap,norm,patches,box\_to\_anchor=(1.65, 1.03))



**HDF 层次化的数据格式（Hierarchical Data Format）**

层次数据格式（Hierarchical Data Format，HDF）是旨在存储和组织大量的数据一组文件格式（HDF4，HDF5），最初由美国国家超级计算应用中心（U.S. National Center for Supercomputing Applications）开发，由HDF集团支持，该集团是一个非营利性公司，其使命是确保HDF5技术的持续发展和以HDF存储数据的持续可及性。HDF数据格式是用于存储MODIS遥感数据的常用格式，可以使用rioxarray库读取。因为HDF文件的层次化，且元数据包含在数据中，因此需要循环读取主数据集和嵌套在主数据集中的子数据集来访问各层数据，例如MCD12Q1v006的不同分类层等。

下述代码读取了MCD12Q1v006原始数据集一个瓦片的数据，并打印读取结果，从Dimensions字段可以查看到波段的大小，为2400×2400；从Coordinates字段可以查看波段的数值类型为int32，x和y坐标值数据类型为float64，空间投影spatial\_ref则可以通过tile\_single.rio.crs方式打印查看；Data variables字段则给出了所有层名称，对应到MCD12Q1\_v006各层分类数据和相关数据层，通过层名可以分别提取各个分类数据；Attributes字段提供了数据集的相关信息。

import rioxarray as rxr  
  
tile\_single=rxr.open\_rasterio(MCD12Q1v006\_group['A2002001'][0], masked=True)  
tile\_single

xarray.Dataset  
Dimensions: (band: 1, x: 2400, y: 2400)  
Coordinates:  
\* band (band) int32 1  
\* x (x) float64 4.448e+06 4.448e+06 ... 5.559e+06 5.56e+06  
\* y (y) float64 6.671e+06 6.671e+06 ... 5.56e+06 5.56e+06  
 spatial\_ref int32 0  
Data variables: (12/13)  
 LC\_Type1 (band, y, x) float32 ...  
 LC\_Prop2 (band, y, x) float32 ...  
 LC\_Prop3 (band, y, x) float32 ...  
 QC (band, y, x) float32 ...  
 LW (band, y, x) float32 ...  
 LC\_Type2 (band, y, x) float32 ...  
 ... ...  
 LC\_Type4 (band, y, x) float32 ...  
 LC\_Type5 (band, y, x) float32 ...  
 LC\_Prop1\_Assessment (band, y, x) float32 ...  
 LC\_Prop2\_Assessment (band, y, x) float32 ...  
 LC\_Prop3\_Assessment (band, y, x) float32 ...  
 LC\_Prop1 (band, y, x) float32 ...  
Attributes: (12/64)  
 ALGORITHMPACKAGEACCEPTANCEDATE: 1998-01-01  
 ALGORITHMPACKAGEMATURITYCODE: LAUNCH  
 ALGORITHMPACKAGENAME: MOD12Q1  
 ALGORITHMPACKAGEVERSION: V2.0  
 ASSOCIATEDINSTRUMENTSHORTNAME.1: MODIS  
 ASSOCIATEDINSTRUMENTSHORTNAME.2: MODIS  
 ... ...  
 SOUTHBOUNDINGCOORDINATE: 50.0  
 SPSOPARAMETERS: 2669  
 TileID: 51022003  
 VERSIONID: 6  
 VERTICALTILENUMBER: 3  
 WESTBOUNDINGCOORDINATE: 62.228953

查看数据的投影信息，可以使用tile\_single.rio.crs，或者tile\_single.spatial\_ref方式，前者仅返回CRS信息，后者则可以返回更多投影相关信息。

tile\_single.rio.crs

CRS.from\_wkt('PROJCS["unnamed",GEOGCS["Unknown datum based upon the custom spheroid",DATUM["Not specified (based on custom spheroid)",SPHEROID["Custom spheroid",6371007.181,0]],PRIMEM["Greenwich",0],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]]],PROJECTION["Sinusoidal"],PARAMETER["longitude\_of\_center",0],PARAMETER["false\_easting",0],PARAMETER["false\_northing",0],UNIT["Meter",1],AXIS["Easting",EAST],AXIS["Northing",NORTH]]')

tile\_single.spatial\_ref

xarray.DataArrays 'spatial\_ref'  
array(0)  
Coordinates:  
 spatial\_ref int32 0  
Attributes: (12/16)  
 crs\_wkt: PROJCS[&quot;unnamed&quot;,GEOGCS[&quot;Unknown datum b...  
 semi\_major\_axis: 6371007.181  
 semi\_minor\_axis: 6371007.181  
 inverse\_flattening: 0.0  
 reference\_ellipsoid\_name: Custom spheroid  
 longitude\_of\_prime\_meridian: 0.0  
 ... ...  
 grid\_mapping\_name: sinusoidal  
 longitude\_of\_projection\_origin: 0.0  
 false\_easting: 0.0  
 false\_northing: 0.0  
 spatial\_ref: PROJCS[&quot;unnamed&quot;,GEOGCS[&quot;Unknown datum b...  
 GeoTransform: 4447802.078667 463.31271652750013 0.0 66...

读取分类数据层，可以用tile\_single.LC\_Prop1或者tile\_single['LC\_Prop1']方式。分类数据层仅显示该层相关信息，从Attributes字段可以查看到分类的名称和对应的栅格值。如果只查看Attributes字段信息，可以执行tile\_single.LC\_Prop1.attrs代码。

tile\_single.LC\_Prop1

xarray.DataArray LC\_Prop1 (band: 1, y: 2400, x: 2400)  
[5760000 values with dtype=float32]  
Coordinates:  
\* band (band) int32 1  
\* x (x) float64 4.448e+06 4.448e+06 ... 5.559e+06 5.56e+06  
\* y (y) float64 6.671e+06 6.671e+06 6.671e+06 ... 5.56e+06 5.56e+06  
 spatial\_ref int32 0  
Attributes: (12/21)  
 Barren: 1  
 Deciduous Broadleaf Forests: 14  
 Deciduous Needleleaf Forests: 13  
 Dense Herbaceous: 31  
 Dense Shrublands: 41  
 Evergreen Broadleaf Forests: 12  
 ... ...  
 Sparse Shrublands: 43  
 Unclassified: 255  
 valid\_range: 1, 43  
 Water Bodies: 3  
 scale\_factor: 1.0  
 add\_offset: 0.0

tile\_single.LC\_Prop1.values方式可以提取波段数值。

tile\_single.LC\_Prop1.values

array([[[31., 31., 22., ..., 21., 21., 21.],  
 [31., 31., 31., ..., 21., 21., 21.],  
 [22., 22., 31., ..., 21., 21., 21.],  
 ...,  
 [31., 32., 32., ..., 32., 32., 32.],  
 [31., 31., 31., ..., 32., 32., 32.],  
 [31., 31., 31., ..., 32., 32., 32.]]], dtype=float32)

下述则打印了MCD12Q1v006合并的数据，因为仅合并了LC\_Prop1层，为LCCS1分类数据。从打印信息可以得知波段大小为y: 12000 x: 26401，Attributes信息同上述单个瓦片的LC\_Prop1层信息。

LCCS1\_merged

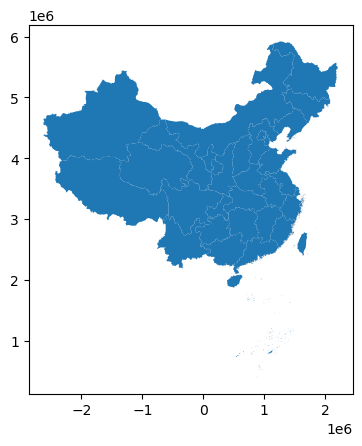
xarray.DataArray LC\_Prop1 (band: 1, y: 12000, x: 26401)  
array([[[15., 15., 15., ..., nan, nan, nan],  
 [15., 15., 15., ..., nan, nan, nan],  
 [15., 15., 15., ..., nan, nan, nan],  
 ...,  
 [nan, nan, nan, ..., 3., 3., nan],  
 [nan, nan, nan, ..., 3., 3., nan],  
 [nan, nan, nan, ..., 3., 3., nan]]], dtype=float32)  
Coordinates:  
\* x (x) float64 3.336e+06 3.337e+06 ... 1.557e+07 1.557e+07  
\* y (y) float64 6.671e+06 6.671e+06 ... 1.113e+06 1.112e+06  
\* band (band) int32 1  
 spatial\_ref int32 0  
Attributes: (12/22)  
 Barren: 1  
 Deciduous Broadleaf Forests: 14  
 Deciduous Needleleaf Forests: 13  
 Dense Herbaceous: 31  
 Dense Shrublands: 41  
 Evergreen Broadleaf Forests: 12  
 ... ...  
 Unclassified: 255  
 valid\_range: 1, 43  
 Water Bodies: 3  
 scale\_factor: 1.0  
 add\_offset: 0.0  
 \_FillValue: nan

合并的原始MCD12Q1v006数据，通常需要重新配置投影并裁切到更明确的分析区域。

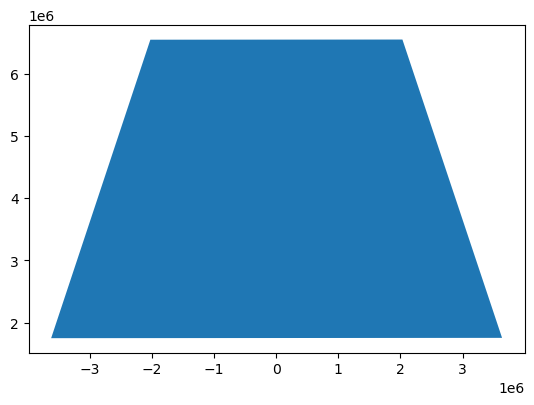
1. 通过定义gdf\_box\_construction()函数，构建具有地理空间坐标和投影，用于裁切的矩形对象；
2. 为了方便地图观察，裁切前先对分类数据通过xarray.rio.reproject方法重投影，投影来自于中国科学院资源环境科学与数据中心的[2015年中国省级行政边界数据](https://www.resdc.cn/data.aspx?DATAID=200)⑪。为了方便投影信息调用，将其存储于AttrDict类方法实例化对象属性字典下；
3. 裁切的方法使用rioxarray库提供的xarray.rio.clip方法，裁切边界外的单元将被nan替换；
4. 因为被裁切的区域填充为nan，为了聚焦分析区域并减小数据量，需要移除全部为空的行或列，定义array2d\_drop\_allNanRowCol()函数实现。

import geopandas as gpd  
ch\_provincial\_borders=gpd.read\_file(args.data.CH\_provincial\_borders\_fn)  
proj\_Krasovsky\_1940\_Albers=ch\_provincial\_borders.crs  
\_\_C.gi.Krasovsky\_1940\_Albers=proj\_Krasovsky\_1940\_Albers  
  
print(args.gi.Krasovsky\_1940\_Albers)  
ch\_provincial\_borders.plot();

PROJCS["Krasovsky\_1940\_Albers",GEOGCS["Unknown datum based upon the Krassowsky 1940 ellipsoid",DATUM["Not\_specified\_based\_on\_Krassowsky\_1940\_ellipsoid",SPHEROID["Krassowsky 1940",6378245,298.3,AUTHORITY["EPSG","7024"]],AUTHORITY["EPSG","6024"]],PRIMEM["Greenwich",0],UNIT["Degree",0.0174532925199433]],PROJECTION["Albers\_Conic\_Equal\_Area"],PARAMETER["latitude\_of\_center",0],PARAMETER["longitude\_of\_center",105],PARAMETER["standard\_parallel\_1",25],PARAMETER["standard\_parallel\_2",47],PARAMETER["false\_easting",0],PARAMETER["false\_northing",0],UNIT["metre",1,AUTHORITY["EPSG","9001"]],AXIS["Easting",EAST],AXIS["Northing",NORTH]]



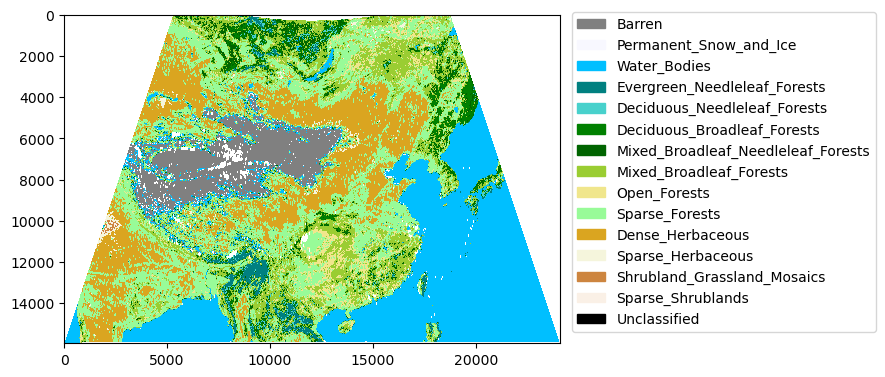
def gdf\_box\_construction(bounding,crs=4326):  
 '''  
 根据指定的经纬度范围建立具有空间坐标和投影的矩形（盒体）GeoDataFrame格式数据对象，通常用于地理数据的裁切提取操作  
  
 Parameters  
 ----------  
 bounding : list(float)  
 边界经纬度坐标，顺序为[minx, miny, maxx, maxy].  
 crs : CRS, optional  
 指定转换的投影坐标，crs形式. The default is 4326.  
  
 Returns  
 -------  
 geodf : GeoDataFrame  
 具有空间坐标和投影的矩形区域.  
  
 '''   
 import geopandas as gpd  
 from shapely.geometry import box  
  
 geodf=gpd.GeoDataFrame(  
 geometry=[  
 box(\*bounding)  
 ],  
 crs=4326  
 )  
 geodf.to\_crs(crs,inplace=True)  
 geodf.plot()   
   
 return geodf  
   
bounding\_gdf=gdf\_box\_construction(bounding=[73.14, 11.56, 137.00, 57.55],crs=args.gi.Krasovsky\_1940\_Albers)



LCCS1\_merged\_KA=LCCS1\_merged.rio.reproject(args.gi.Krasovsky\_1940\_Albers)   
LCCS1\_merged\_KA\_clipped=LCCS1\_merged\_KA.rio.clip(bounding\_gdf.geometry.values, bounding\_gdf.crs, drop=False)merged\_reproject\_clipped

def array2d\_drop\_allNanRowCol(array,drop\_rows=True,drop\_cols=True):  
 '''  
 移除2D数组全空行或列   
  
 Parameters  
 ----------  
 array : array  
 2D数组.  
 drop\_rows : bool, optional  
 如果为True，则移除全空行. The default is True.  
 drop\_cols : bool, optional  
 如果为True，则移除全空列. The default is True.  
  
 Returns  
 -------  
 array  
 移除全空行或列的数组.  
  
 '''   
 import numpy as np  
   
 if drop\_rows and drop\_cols:  
 return array[~np.all(np.isnan(array),axis=1),~np.all(np.isnan(array),axis=0)]  
 elif drop\_rows and not drop\_cols:  
 return array[~np.all(np.isnan(array),axis=1),:]  
 elif drop\_cols and not drop\_rows:  
 return array[:,~np.all(np.isnan(array),axis=0)]  
  
LCCS1\_merged\_KA\_clipped\_notna=array2d\_drop\_allNanRowCol(LCCS1\_merged\_KA\_clipped[0])

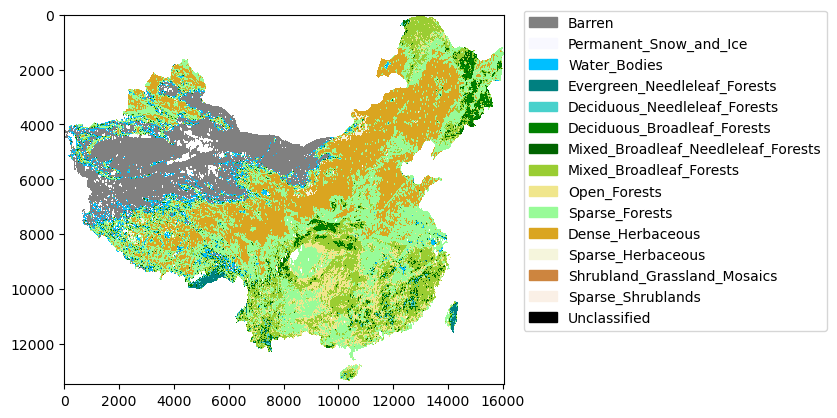
import matplotlib.pyplot as plt  
plt.imshow(LCCS1\_merged\_KA\_clipped\_notna, cmap=cmap, norm=norm)  
plt.legend(handles=patches, bbox\_to\_anchor=(1.65, 1.03),ncol=1, fancybox=True,);



上述使用了自定义矩形裁切边界裁切数据，如果仅保留中国国界内的数据，则可以使用ch\_provincial\_borders数据进行裁切，并且使用自定义函数array2d\_drop\_allNanRowCol()再次移除全空行或列。

LCCS1\_merged\_KA\_clipped\_CH=LCCS1\_merged\_KA\_clipped\_notna.rio.clip(ch\_provincial\_borders.geometry.values, ch\_provincial\_borders.crs, drop=False)  
LCCS1\_merged\_KA\_clipped\_notna\_CH=array2d\_drop\_allNanRowCol(LCCS1\_merged\_KA\_clipped\_CH)

plt.imshow(LCCS1\_merged\_KA\_clipped\_notna\_CH, cmap=cmap, norm=norm)  
plt.legend(handles=patches, bbox\_to\_anchor=(1.75, 1.03),ncol=1, fancybox=True,);



**xarray.DataArray 数据保存**

rioxarray库是rasterio库对[Xarray](https://docs.xarray.dev/en/stable/index.html)⑫的扩展。Xarray是一个用于处理标记多维（N-dimensional, ND）数组的Python库，且包括高级分析和可视化等功能。Pandas库的数据结构包括Series和DataFramd两种，为1维度和2维度；NumPy库不限维度，处理多维数组，但是每一数组只能有一种数据类型，因此Xarray库试图解决多维数据且具有不同数据类型的问题：

1. Xarray库使用命名的维度来代替轴标签，使得数据选择和维度操作更容易；
2. Xarray库支持多种类型数据，可以在一个ND数组中保存异质数据；
3. obj.attrs方法可以追踪对象任意元数据。

Xarray有两类数据结构，为DataArray和DataSet。DataArray用于单个数据变量，DataSet为多个DataArray数据变量的容器。

上述合并、裁切后的LCCS1\_merged\_KA\_clipped\_notna\_CH数据，为DataArray格式数据，Xarray库提供了多种保存的方法，下述代码采用了两种方式，一种是ojb.rio.to\_raster方法，将DataArray对象存储为TIFF格式数据；另一种是obj.to\_netcdf方法，将DataArray对象存储为[netCDF（Network Common Data Form）](https://en.wikipedia.org/wiki/NetCDF)⑬格式文件。netCDF是一种用于创建、存储、访问和共享面向多维科学数据的文件格式，具有自我描述性（self-describing）。TIFF格式数据仅为栅格数据，并不包括属性等说明，而netCDF的自我描述性使得数据存储的信息更加全面。保存为netCDF数据文件，仍可以使用rioxarray库的open\_rasterio方法读取。

在执行obj.to\_netcdf时，如果属性值等命名不符合要求，则会提示错误，例如元数据中的属性值，即分类字典中包含有/ 等非法字符，例如’Mixed Broadleaf/Needleleaf Forests’，因此需要替换该字符。为了避免直接修改原数据，可以使用obj.copy()复制为一个副本。

save\_temp\_tif\_fn=r"G:\data\MCD12Q1v006\temp\LCCS1\_merged\_KA\_clipped\_notna\_CH.tif"  
LCCS1\_merged\_KA\_clipped\_notna\_CH.rio.to\_raster(save\_temp\_tif\_fn)

LCCS1\_merged\_KA\_clipped\_notna\_CH\_copy=LCCS1\_merged\_KA\_clipped\_notna\_CH.copy()  
LCCS1\_merged\_KA\_clipped\_notna\_CH\_copy\_attrs=LCCS1\_merged\_KA\_clipped\_notna\_CH\_copy.attrs  
LCCS1\_merged\_KA\_clipped\_notna\_CH\_copy\_attrs\_update={k.replace("/","\_"):v for k,v in LCCS1\_merged\_KA\_clipped\_notna\_CH\_copy\_attrs.items()}  
LCCS1\_merged\_KA\_clipped\_notna\_CH\_copy.attrs=LCCS1\_merged\_KA\_clipped\_notna\_CH\_copy\_attrs\_update  
LCCS1\_merged\_KA\_clipped\_notna\_CH\_copy

xarray.DataArray LC\_Prop1 (y: 13477, x: 16046)  
array([[nan, nan, nan, ..., nan, nan, nan],  
 [nan, nan, nan, ..., nan, nan, nan],  
 [nan, nan, nan, ..., nan, nan, nan],  
 ...,  
 [nan, nan, nan, ..., nan, nan, nan],  
 [nan, nan, nan, ..., nan, nan, nan],  
 [nan, nan, nan, ..., nan, nan, nan]], dtype=float32)  
Coordinates:  
\* x (x) float64 -2.626e+06 -2.626e+06 ... 2.207e+06 2.207e+06  
\* y (y) float64 5.922e+06 5.921e+06 ... 1.753e+06 1.753e+06  
 band int32 1  
 spatial\_ref int32 0  
Attributes: (12/22)  
 Barren: 1  
 Deciduous Broadleaf Forests: 14  
 Deciduous Needleleaf Forests: 13  
 Dense Herbaceous: 31  
 Dense Shrublands: 41  
 Evergreen Broadleaf Forests: 12  
 ... ...  
 Unclassified: 255  
 valid\_range: 1, 43  
 Water Bodies: 3  
 scale\_factor: 1.0  
 add\_offset: 0.0  
 \_FillValue: nan

save\_temp\_hdf\_fn=r"G:\data\MCD12Q1v006\temp\LCCS1\_merged\_KA\_clipped\_notna\_CH.nc"  
LCCS1\_merged\_KA\_clipped\_notna\_CH\_copy.to\_netcdf(save\_temp\_hdf\_fn)

**批量处理和保存MCD12Q1\_v006 各年数据**

批量处理MCD12Q1\_v006各年数据，将上述合并、重投影、裁切和保存的过程置于一个函数MCD12Q1v006\_batch\_processing()之内，简化操作流程。保存的数据类型为netCDF格式文件，通过配置group参数可以将各年的数据存储于同一文件下，避免不方便管理的零散数据文件。合并存储后的数据文件（中国区域）约17G大小，如果单个存储文件过大，可以存储时分批存储于不同的文件下，减少单个文件的大小。

def MCD12Q1v006\_batch\_processing(fns\_group,clip\_border,layer,save\_fn):  
 '''  
 批量处理MCD12Q1v006各年数据，包括合并、重投影、裁切，保存等过程。投影将转换为裁切对象投影。保存数据文件类型为netcdf，并配置`group`参数，将合并后的各年数据存于同一个文件下  
  
 Parameters  
 ----------  
 fns\_group : dict(string:string)  
 由data\_grouping\_basedon\_formatting\_pattern()函数计算获取，为按年份分组后的数据集.  
 clip\_border : GeoDataFrame  
 用于裁切的Polygon对象.  
 layer : string  
 MCD12Q1v006分类层.  
 save\_fn : string  
 保存文件路径，后缀名通常为'.nc'.  
  
 Returns  
 -------  
 None.  
  
 '''   
 from rioxarray.merge import merge\_arrays  
 import rioxarray as rxr  
 import numpy as np  
 from tqdm import tqdm  
  
 for g in tqdm(fns\_group):  
 arrays=[rxr.open\_rasterio(i,masked=True)[layer] for i in fns\_group[g]]  
 merged=merge\_arrays(arrays)  
 merged\_reproject=merged.rio.reproject(clip\_border.crs)   
 merged\_reproject\_clipped=merged\_reproject.rio.clip(clip\_border.geometry.values, clip\_border.crs, drop=False)  
 array=merged\_reproject\_clipped[0]  
 merged\_reproject\_clipped\_notna=array[~np.all(np.isnan(array),axis=1),~np.all(np.isnan(array),axis=0)]  
 attrs=merged\_reproject\_clipped\_notna.attrs  
 attrs\_update={k.replace("/","\_"):v for k,v in attrs.items()}  
 merged\_reproject\_clipped\_notna.attrs=attrs\_update  
   
 merged\_reproject\_clipped\_notna.to\_netcdf(save\_fn,"a",group=g,engine="netcdf4") # engine="netcdf4"   
  
MCD12Q1v006\_batch\_processing(MCD12Q1v006\_group,ch\_provincial\_borders,'LC\_Prop1',save\_fn=args.data.MCD12Q1v006\_preprocessed)

100%|██████████| 20/20 [32:28<00:00, 97.45s/it]

读取给定group参数保存的netcdf数据格式文件，在读取时使用rioxarray.open\_rasterio方法或xarray库的xarray.open\_dataset方法，但是需要给出组名参数group，例如rxr.open\_rasterio(args.data.MCD12Q1v006\_preprocessed, group='A2001001')。为了按年顺序读取文件，提取组名中时间信息转换为datetime对象并按其排序构建列表。

from datetime import datetime  
  
g\_names=MCD12Q1v006\_group.keys()  
g\_names\_dt=[(g,datetime.strptime(g[1:5],'%Y')) for g in g\_names]  
g\_names\_dt.sort(key=lambda x:x[1])  
g\_names\_dt

[('A2001001', datetime.datetime(2001, 1, 1, 0, 0)),  
 ('A2002001', datetime.datetime(2002, 1, 1, 0, 0)),  
 ('A2003001', datetime.datetime(2003, 1, 1, 0, 0)),  
 ('A2004001', datetime.datetime(2004, 1, 1, 0, 0)),  
 ('A2005001', datetime.datetime(2005, 1, 1, 0, 0)),  
 ('A2006001', datetime.datetime(2006, 1, 1, 0, 0)),  
 ('A2007001', datetime.datetime(2007, 1, 1, 0, 0)),  
 ('A2008001', datetime.datetime(2008, 1, 1, 0, 0)),  
 ('A2009001', datetime.datetime(2009, 1, 1, 0, 0)),  
 ('A2010001', datetime.datetime(2010, 1, 1, 0, 0)),  
 ('A2011001', datetime.datetime(2011, 1, 1, 0, 0)),  
 ('A2012001', datetime.datetime(2012, 1, 1, 0, 0)),  
 ('A2013001', datetime.datetime(2013, 1, 1, 0, 0)),  
 ('A2014001', datetime.datetime(2014, 1, 1, 0, 0)),  
 ('A2015001', datetime.datetime(2015, 1, 1, 0, 0)),  
 ('A2016001', datetime.datetime(2016, 1, 1, 0, 0)),  
 ('A2017001', datetime.datetime(2017, 1, 1, 0, 0)),  
 ('A2018001', datetime.datetime(2018, 1, 1, 0, 0)),  
 ('A2019001', datetime.datetime(2019, 1, 1, 0, 0)),  
 ('A2020001', datetime.datetime(2020, 1, 1, 0, 0))]

在存储为netcdf数据格式文件时，保留了原始MCD12Q1\_v006数据的信息，如果直接打印MCD12Q1v006\_2001时，会提示有Data variables变量层为LC\_Prop1，因此如果只读取变量，执行MCD12Q1v006\_2001.LC\_Prop1即可。

import rioxarray as rxr  
MCD12Q1v006\_2001=rxr.open\_rasterio(args.data.MCD12Q1v006\_preprocessed, group='A2001001')  
MCD12Q1v006\_2001.LC\_Prop1

xarray.DataArray LC\_Prop1 (band: 1, y: 14313, x: 16045)  
[229652085 values with dtype=float32]  
Coordinates:  
\* band (band) int32 1  
\* x (x) float64 -2.626e+06 -2.626e+06 ... 2.206e+06 2.207e+06  
\* y (y) float64 5.922e+06 5.921e+06 ... 1.049e+06 1.048e+06  
 spatial\_ref int32 0  
Attributes: (12/75)  
 /A2001001/LC\_Prop1#add\_offset: 0  
 /A2001001/LC\_Prop1#Barren: 1  
 /A2001001/LC\_Prop1#coordinates: band  
 /A2001001/LC\_Prop1#Deciduous Broadleaf Forests: 14  
 /A2001001/LC\_Prop1#Deciduous Needleleaf Forests: 13  
 /A2001001/LC\_Prop1#Dense Herbaceous: 31  
 ... ...  
 Sparse Herbaceous: 32  
 Sparse Shrublands: 43  
 Unclassified: 255  
 valid\_range: 1, 43  
 Water Bodies: 3  
 \_FillValue: nan

除了保存为netcdf等格式文件外，可以直接指定DataArray数据变量层转换为.tif格式的栅格数据，并保存到指定文件夹下。

def MCD12Q1v006\_gnc2tif(g\_names\_dt,nc\_fn,layer,save\_path):  
 '''  
 按组提取netcdf格式数据，并转换为.tif格式的栅格数据  
  
 Parameters  
 ----------  
 g\_names\_dt : dict(string:datetime)  
 包含组名和对应时间信息的列表，例如：  
 [('A2001001', datetime.datetime(2001, 1, 1, 0, 0)),  
 ('A2002001', datetime.datetime(2002, 1, 1, 0, 0))].  
 nc\_fn : string  
 含组的netcdf格式文件路径名.  
 layer : string  
 转换为栅格的变量名（Data variables）.  
 save\_path : string  
 保存路径根目录.  
  
 Returns  
 -------  
 fns : list(string)  
 保存的栅格文件路径.  
  
 '''   
 from tqdm import tqdm  
   
 fns=[]  
 for g\_n,dt in tqdm(g\_names\_dt):  
 g=rxr.open\_rasterio(nc\_fn, group=g\_n)   
 print(f"{dt.year}.tif")  
 fn=os.path.join(save\_path,f"{dt.year}.tif")  
 fns.append(fn)  
 g\_layer=g[layer]  
 g\_layer.rio.to\_raster(fn)   
   
 return fns  
   
MCD12Q1v006\_tif\_fns=MCD12Q1v006\_gnc2tif(g\_names\_dt,args.data.MCD12Q1v006\_preprocessed,'LC\_Prop1',args.data.MCD12Q1v006\_tif)

100%|██████████| 20/20 [00:00<?, ?it/s]

**按采样点提取数据**

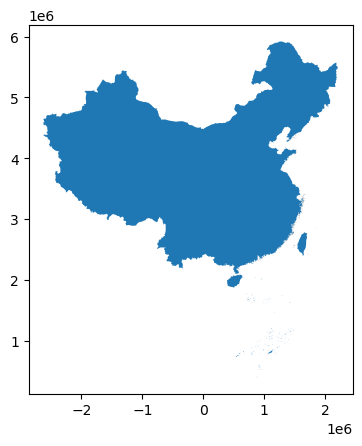
原始历年MCD12Q1\_v006数据各层读取为DataArray格式，虽然具有统一的大地坐标和投影，但是坐标值Coordinates不同年份可能会存在差异，因此通过xarray.DataArray.sel方法按照统一坐标列表提取历年各层值（即栅格值）会存在偏差，造成数据错误；同时，中国区数据量为(y: 13477 x: 16046）即多个栅格单元或坐标，且后续进行经典离散时间马尔可夫链分析时，只需要足够多的单元样本就可进行，因此采用生成采样点，由采样点提取历年栅格单元值的方法。

定义random\_pts\_in\_bounds()和random\_pts\_in\_geoBounds()函数，实现给定边界生成对应投影坐标GeoDataFrame格式的采样点。给定的边界为GeoDataFrame格式的Polygon对象，如果使用中国区2015年中国省级行政边界数据，则需要用gdf.dissolve方法合并各行数据为一个Polygon对象。对于不规则图形边界，生成的采样点是边界外接矩形指定数量的采样点，当仅保留边界内的点时，采样点数据会变少。

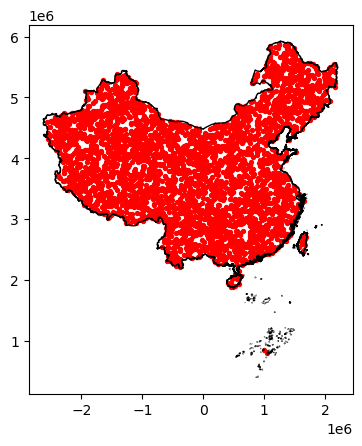
ch\_provincial\_borders\_dissolved=ch\_provincial\_borders.dissolve()  
ch\_provincial\_borders\_dissolved

|  | **geometry** | **AREA** | **PERIMETER** | **SHENG\_** | **SHENG\_ID** | **SHENG** | **name** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | MULTIPOLYGON (((494657.361 1879931.569, 494643... | 4.525440e+11 | 6666710.0 | 2 | 23 | 23 | 黑龙江 |

ch\_provincial\_borders\_dissolved.plot()



def random\_pts\_in\_bounds(polygon\_gdf, number):   
 '''  
 根据给定的Polygon格式边界，生成指定数量的随机点坐标值  
  
 Parameters  
 ----------  
 polygon\_gdf : GeoDataFrame  
 Polygon边界.  
 number : int  
 生成随机点坐标值数量.  
  
 Returns  
 -------  
 x : array(float)  
 X坐标值.  
 y : array(float)  
 Y坐标值.  
  
 '''   
 import numpy as np  
 from shapely.geometry import Polygon  
   
 minx, miny, maxx, maxy=polygon\_gdf.bounds.values[0]  
 x=np.random.uniform( minx, maxx, number )  
 y=np.random.uniform( miny, maxy, number )  
 return x, y  
  
def random\_pts\_in\_geoBounds(polygon\_gdf,number,plot=False):  
 '''  
 根据给定的Polygon格式边界，生成指定数量对应到Polygon对象投影的GeoDataFrame格式随机点，配合random\_pts\_in\_bounds()函数  
  
 Parameters  
 ----------  
 polygon\_gdf : GeoDataFrame  
 Polygon边界.  
 number : int  
 生成随机点的数量.  
 plot : bool, optional  
 是否打印边界和随机点地图查看. The default is False.  
  
 Returns  
 -------  
 pts\_in\_polygon\_gdf : TYPE  
 DESCRIPTION.  
  
 '''   
 from shapely.geometry import Point  
 import geopandas as gpd  
 import matplotlib.pyplot as plt  
   
 x,y=random\_pts\_in\_bounds(polygon\_gdf, number)  
 pts\_df=pd.DataFrame()  
 pts\_df['points']=list(zip(x,y))   
 pts\_df['points']=pts\_df['points'].apply(Point)  
 points\_gdf=gpd.GeoDataFrame(pts\_df, geometry='points')  
 points\_gdf.set\_crs(polygon\_gdf.crs,inplace=True)  
  
 Sjoin=gpd.tools.sjoin(points\_gdf,polygon\_gdf,predicate="within", how='left')  
 idx\_name=polygon\_gdf.index.values[0]  
 pts\_in\_polygon\_gdf=points\_gdf[Sjoin.index\_right==idx\_name]  
   
 if plot:  
 ax=polygon\_gdf.boundary.plot(linewidth=1, edgecolor="black")  
 pts\_in\_polygon\_gdf.plot(ax=ax, linewidth=1, color="red", markersize=8)  
 plt.show()  
   
 return pts\_in\_polygon\_gdf   
  
pts\_num=10\_000   
pts\_in\_polygon\_gdf=random\_pts\_in\_geoBounds(ch\_provincial\_borders\_dissolved, pts\_num,plot=True)



定义extract\_raster\_vals\_at\_pts()和extract\_raster\_vals\_at\_pts\_batch()函数根据采样点坐标值列表提取对应栅格单元值。提取单元值的主要方法是使用rasterio库的sample方法。

sample\_pt\_coords=[(p.x,p.y) for p in pts\_in\_polygon\_gdf.points.values]

def extract\_raster\_vals\_at\_pts(pt\_coords,raster\_fn):  
 '''  
 对一个栅格数据执行给定采样点提取栅格值  
  
 Parameters  
 ----------  
 pt\_coords : list(float)  
 采样点坐标值列表，格式为[(x1,y1),(x2,y2)].  
 raster\_fn : string  
 栅格文件路径名.  
  
 Returns  
 -------  
 sample\_vals : array  
 采样点位置栅格值.  
  
 '''   
 import rasterio as rio  
 import numpy as np  
   
 raster=rio.open(raster\_fn)  
 sample\_vals=np.stack(list(raster.sample(pt\_coords)))  
   
 return sample\_vals  
  
def extract\_raster\_vals\_at\_pts\_batch(pt\_coords,raster\_fns\_lst):  
 '''  
 对多个栅格数据执行给定采样点提取栅格值  
  
 Parameters  
 ----------  
 pt\_coords : list(float)  
 采样点坐标值列表，格式为[(x1,y1),(x2,y2)].  
 raster\_fns\_lst : list(string)  
 多个栅格文件路径名列表.  
  
 Returns  
 -------  
 sample\_vals\_array : array  
 多个栅格采样点值数组.  
 idx\_lst : list(int)  
 栅格数索引值列表.  
  
 '''   
 from tqdm.notebook import tqdm  
   
 idx\_lst=[]  
 sample\_vals\_lst=[]  
 pbar=tqdm(total=len(raster\_fns\_lst))  
 for i,fn in tqdm(enumerate(raster\_fns\_lst)):  
 sample\_vals=extract\_raster\_vals\_at\_pts(pt\_coords,fn)  
 idx\_lst.append(i)  
 sample\_vals\_lst.append(sample\_vals)  
 pbar.update(1)  
   
 pbar.close()   
 sample\_vals\_array=np.concatenate(sample\_vals\_lst, axis=1)  
 return sample\_vals\_array,idx\_lst   
   
sample\_vals\_array,idx\_lst=extract\_raster\_vals\_at\_pts\_batch(sample\_pt\_coords,MCD12Q1v006\_tif\_fns)

0%| | 0/20 [00:00<?, ?it/s]  
  
  
  
0it [00:00, ?it/s]

保存按采样点提取的栅格值数组，方便后续实验读取。数组的第1维度对应采样点，第2维度对应历年栅格。

import pickle  
with open(args.data.MCD12Q1v006\_years\_vals\_array,'wb') as f:  
 pickle.dump(sample\_vals\_array,f)

sample\_vals\_array

array([[31., 31., 31., ..., 31., 31., 31.],  
 [ 1., 1., 1., ..., 1., 1., 1.],  
 [ 1., 1., 1., ..., 1., 1., 1.],  
 ...,  
 [31., 31., 31., ..., 31., 31., 31.],  
 [ 1., 1., 1., ..., 1., 1., 1.],  
 [22., 22., 22., ..., 22., 22., 22.]], dtype=float32)

#### 2）芝加哥市发生报告的犯罪事件数据集

从[Chicago Data Portal](https://data.cityofchicago.org/)获取[Crimes - 2001 to present](https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present-Dashboard/5cd6-ry5g)⑭数据集，该数据集提取自芝加哥警察局的CLEAR（Citizen Law Enforcement Analysis and Reporting，公民执法分析和报告）系统，反映了2001年至今在芝加哥市发生报告的犯罪事件。时空类数据除了空间分布的位置信息外，包括时间轴向上的属性变化数据，符合在时间序列上取多个截面，在这些截面上同时选取样本观测值构成的面板数据（Panel data）。该数据包括犯罪地点（经纬度值），也包括该地点所属社区区域编号，最终要分析的内容是按给定时间区间段重采样各个社区范围内累计报告犯罪数量在时间上和在时空上的动态变化，因此数据预处理的内容包括：

1. 将没有空间几何字段报告的犯罪记录数据按社区编号字段Community Area合并入同样下载于Chicago Data Portal的“Boundaries - Community Areas (current)”数据（含有社区区域地理空间信息的社区范围），对应Community Area的字段为area\_numbe；
2. 因为时空面板数据表述为GeoDataFrame格式的二维度数据，在列中既包括样本所属空间几何对象（即按社区分组），也包括时间轴向上的日期数据及对应日期的各属性数据，需按组（社区范围）统计给定时间长度报告犯罪数累积量，即按组按给定时间长度重采样;
3. 为方便时间序列数据的处理，将含有时间的字符串转换为统一格式化的Datatime格式数据。

from database import postSQL2gpd,gpd2postSQL  
  
\_\_C.data.crimes='G:/data/Crimes\_-\_2001\_to\_Present.csv' # 芝加哥城犯罪记录数据  
\_\_C.data.community\_areas="./data/ChicagoCommunityAreas/ChicagoCommunityAreas.shp" # 芝加哥城社区边界数据  
\_\_C.data.crimes\_resample='G:/data/crimes\_resample.gpkg' # 用于存储时间采样后的数据

使用pandas库的read\_csv方法直接读取CSV格式报告的犯罪记录数据集，打印一行数据查看含有的字段名和对应的值，确定样本所属社区编号列为Community Area，发生时间列为Date。

import pandas as pd  
crimes\_df=pd.read\_csv(args.data.crimes)

print(crimes\_df.loc[0])  
crimes\_df.head()

ID 10224738  
Case Number HY411648  
Date 09/05/2015 01:30:00 PM  
Block 043XX S WOOD ST  
IUCR 0486  
Primary Type BATTERY  
Description DOMESTIC BATTERY SIMPLE  
Location Description RESIDENCE  
Arrest False  
Domestic True  
Beat 924  
District 9.0  
Ward 12.0  
Community Area 61.0  
FBI Code 08B  
X Coordinate 1165074.0  
Y Coordinate 1875917.0  
Year 2015  
Updated On 02/10/2018 03:50:01 PM  
Latitude 41.815117  
Longitude -87.67  
Location (41.815117282, -87.669999562)  
Historical Wards 2003-2015 29.0  
Zip Codes 14924.0  
Community Areas 59.0  
Census Tracts 706.0  
Wards 3.0  
Boundaries - ZIP Codes 37.0  
Police Districts 23.0  
Police Beats 108.0  
Name: 0, dtype: object

|  | **ID** | **Case Number** | **Date** | **Block** | **IUCR** | **Primary Type** | **Description** | **Location Description** | **Arrest** | **Domestic** | **...** | **Longitude** | **Location** | **Historical Wards 2003-2015** | **Zip Codes** | **Community Areas** | **Census Tracts** | **Wards** | **Boundaries - ZIP Codes** | **Police Districts** | **Police Beats** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 10224738 | HY411648 | 09/05/2015 01:30:00 PM | 043XX S WOOD ST | 0486 | BATTERY | DOMESTIC BATTERY SIMPLE | RESIDENCE | False | True | ... | -87.670000 | (41.815117282, -87.669999562) | 29.0 | 14924.0 | 59.0 | 706.0 | 3.0 | 37.0 | 23.0 | 108.0 |
| **1** | 10224739 | HY411615 | 09/04/2015 11:30:00 AM | 008XX N CENTRAL AVE | 0870 | THEFT | POCKET-PICKING | CTA BUS | False | False | ... | -87.765400 | (41.895080471, -87.765400451) | 4.0 | 4299.0 | 26.0 | 562.0 | 45.0 | 5.0 | 25.0 | 67.0 |
| **2** | 11646166 | JC213529 | 09/01/2018 12:01:00 AM | 082XX S INGLESIDE AVE | 0810 | THEFT | OVER $500 | RESIDENCE | False | True | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **3** | 10224740 | HY411595 | 09/05/2015 12:45:00 PM | 035XX W BARRY AVE | 2023 | NARCOTICS | POSS: HEROIN(BRN/TAN) | SIDEWALK | True | False | ... | -87.716650 | (41.937405765, -87.716649687) | 15.0 | 21538.0 | 22.0 | 216.0 | 12.0 | 39.0 | 7.0 | 168.0 |
| **4** | 10224741 | HY411610 | 09/05/2015 01:00:00 PM | 0000X N LARAMIE AVE | 0560 | ASSAULT | SIMPLE | APARTMENT | False | True | ... | -87.755121 | (41.881903443, -87.755121152) | 11.0 | 22216.0 | 26.0 | 696.0 | 23.0 | 32.0 | 25.0 | 81.0 |

5 rows × 30 columns

使用GeoPandas库的read\_file方法直接读取SHP格式的社区范围数据，打印查看包含列，确定社区编号字段为area\_numbe，几何列为geometry(Polygon几何类型)。

import geopandas as gpd  
community\_area\_gdf=gpd.read\_file(args.data.community\_areas)  
community\_area\_gdf.area\_numbe=community\_area\_gdf.area\_numbe.astype("int64")  
community\_area\_gdf.sort\_values(by=["area\_numbe"],inplace=True)  
community\_area\_gdf.head()

|  | **area** | **area\_num\_1** | **area\_numbe** | **comarea** | **comarea\_id** | **community** | **perimeter** | **shape\_area** | **shape\_len** | **geometry** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **9** | 0.0 | 1 | 1 | 0.0 | 0.0 | ROGERS PARK | 0.0 | 5.125990e+07 | 34052.397576 | POLYGON ((-87.65456 41.99817, -87.65574 41.998... |
| **19** | 0.0 | 2 | 2 | 0.0 | 0.0 | WEST RIDGE | 0.0 | 9.842909e+07 | 43020.689458 | POLYGON ((-87.68465 42.01948, -87.68464 42.019... |
| **30** | 0.0 | 3 | 3 | 0.0 | 0.0 | UPTOWN | 0.0 | 6.509564e+07 | 46972.794555 | POLYGON ((-87.64102 41.95480, -87.64400 41.954... |
| **5** | 0.0 | 4 | 4 | 0.0 | 0.0 | LINCOLN SQUARE | 0.0 | 7.135233e+07 | 36624.603085 | POLYGON ((-87.67441 41.97610, -87.67440 41.976... |
| **47** | 0.0 | 5 | 5 | 0.0 | 0.0 | NORTH CENTER | 0.0 | 5.705417e+07 | 31391.669754 | POLYGON ((-87.67336 41.93234, -87.67342 41.932... |

在“空间自相关分析”部分定义有df\_linking\_geometry()函数，用于包含有属性的DataFrame格式数据合并包含有地理空间信息的GeoDataFrame格式数据，将其置入util\_misc模块调用合并报告的犯罪记录数据和社区范围数据。

import util\_misc  
crimes\_gdf=util\_misc.df\_linking\_geometry(crimes\_df,community\_area\_gdf,key\_df="Community Area",key\_gdf="area\_numbe")

使用pandas库提供的to\_datetime方法将含有时间字符串Date列转换为datatime格式数据并置于新列ts下。

import datetime  
crimes\_gdf['ts']=pd.to\_datetime(crimes\_gdf['Date'])

将时间列数据用于行索引值，为避免名称冲突，将时间行索引名重命名为ts\_idx，避免后续相关执行代码可能发生的错误。

crimes\_gdf.set\_index('ts',inplace=True,drop=False)  
crimes\_gdf.rename\_axis(index={'ts':'ts\_idx'},inplace=True)

通过直接打印GeoDataFrame格式数据，或者只打印前几行和后几行来初步判断时间周期，起始时间为“2001-01-01”，结束时间为“2022-11-22 23:58:00”。

crimes\_gdf.sort\_index(inplace=True)  
crimes\_gdf.head(3)

|  | **area\_numbe** | **geometry** | **ID** | **Case Number** | **Date** | **Block** | **IUCR** | **Primary Type** | **Description** | **Location Description** | **...** | **Location** | **Historical Wards 2003-2015** | **Zip Codes** | **Community Areas** | **Census Tracts** | **Wards** | **Boundaries - ZIP Codes** | **Police Districts** | **Police Beats** | **ts** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ts\_idx** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2001-01-01** | 68 | POLYGON ((-87.62826 41.78316, -87.62826 41.783... | 3723059 | HK830095 | 01/01/2001 12:00:00 AM | 071XX S UNION AVE | 0840 | THEFT | FINANCIAL ID THEFT: OVER $300 | APARTMENT | ... | (41.765053677, -87.641686171) | 31.0 | 21559.0 | 66.0 | 511.0 | 32.0 | 11.0 | 17.0 | 214.0 | 2001-01-01 |
| **2001-01-01** | 17 | POLYGON ((-87.77621 41.93845, -87.77677 41.938... | 9959537 | HY141743 | 01/01/2001 12:00:00 AM | 034XX N NOTTINGHAM AVE | 1752 | OFFENSE INVOLVING CHILDREN | AGG CRIM SEX ABUSE FAM MEMBER | RESIDENCE | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 2001-01-01 |
| **2001-01-01** | 67 | POLYGON ((-87.65487 41.79417, -87.65487 41.794... | 5091512 | HM695128 | 01/01/2001 12:00:00 AM | 057XX S JUSTINE ST | 0840 | THEFT | FINANCIAL ID THEFT: OVER $300 | OTHER | ... | (41.789165627, -87.663083724) | 19.0 | 22257.0 | 65.0 | 384.0 | 2.0 | 23.0 | 17.0 | 266.0 | 2001-01-01 |

3 rows × 33 columns

crimes\_gdf.tail(3)

|  | **area\_numbe** | **geometry** | **ID** | **Case Number** | **Date** | **Block** | **IUCR** | **Primary Type** | **Description** | **Location Description** | **...** | **Location** | **Historical Wards 2003-2015** | **Zip Codes** | **Community Areas** | **Census Tracts** | **Wards** | **Boundaries - ZIP Codes** | **Police Districts** | **Police Beats** | **ts** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ts\_idx** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2022-11-22 23:51:00** | 20 | POLYGON ((-87.73146 41.93173, -87.73139 41.931... | 12900924 | JF484559 | 11/22/2022 11:51:00 PM | 020XX N PULASKI RD | 0460 | BATTERY | SIMPLE | GAS STATION | ... | (41.917425774, -87.726571455) | 27.0 | 22535.0 | 21.0 | 76.0 | 49.0 | 2.0 | 6.0 | 187.0 | 2022-11-22 23:51:00 |
| **2022-11-22 23:53:00** | 25 | POLYGON ((-87.78942 41.91751, -87.78927 41.917... | 12901051 | JF484599 | 11/22/2022 11:53:00 PM | 049XX W CONGRESS PKWY | 0454 | BATTERY | AGGRAVATED P.O. - HANDS, FISTS, FEET, NO / MIN... | RESIDENCE - PORCH / HALLWAY | ... | (41.874030213, -87.748083613) | 36.0 | 22216.0 | 26.0 | 69.0 | 7.0 | 32.0 | 25.0 | 137.0 | 2022-11-22 23:53:00 |
| **2022-11-22 23:58:00** | 23 | POLYGON ((-87.69157 41.88820, -87.69501 41.888... | 12900998 | JF484554 | 11/22/2022 11:58:00 PM | 032XX W GRAND AVE | 0560 | ASSAULT | SIMPLE | STREET | ... | (41.898771348, -87.707812918) | 41.0 | 4299.0 | 24.0 | 150.0 | 46.0 | 5.0 | 16.0 | 65.0 | 2022-11-22 23:58:00 |

定义按时间长度分组重采样的df\_group\_resample()函数。重采样部分主要使用pandas提供的resample方法，其中rule参数为被赋予特定时间序列频率的字符串别名，别名含义可以从pandas手册[Time series / date functionality：Offset aliases](https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html#offset-aliases)⑮处查看，摘录如下：

| Alias | Description |
| --- | --- |
| B | business day frequency |
| C | custom business day frequency |
| D | calendar day frequency |
| W | weekly frequency |
| M | month end frequency |
| SM | semi-month end frequency (15th and end of month) |
| BM | business month end frequency |
| CBM | custom business month end frequency |
| MS | month start frequency |
| SMS | semi-month start frequency (1st and 15th) |
| BMS | business month start frequency |
| CBMS | custom business month start frequency |
| Q | quarter end frequency |
| BQ | business quarter end frequency |
| QS | quarter start frequency |
| BQS | business quarter start frequency |
| A, Y | year end frequency |
| BA, BY | business year end frequency |
| AS, YS | year start frequency |
| BAS, BYS | business year start frequency |
| BH | business hour frequency |
| H | hourly frequency |
| T, min | minutely frequency |
| S | secondly frequency |
| L, ms | milliseconds |
| U, us | microseconds |
| N | nanoseconds |

def df\_group\_resample(df,val\_column,time\_column,rules,methods=["mean","min","max","sum"],group\_column=None,geometry\_column=None):  
 '''  
 时空数据（面板数据），按照给定的分组，时间长度，数值计算方法重采样数值列。  
  
 Parameters  
 ----------  
 df : DataFrame  
 时空数据.  
 val\_column : string  
 用于重采样的数据值.  
 time\_column : string  
 时间列.  
 rules : string  
 偏移量（时间长度），例'H'，'D'，`W`，`M`，`Y`，'30S'，`3T`，`Q`，`17min`等.  
 methods : list(string), optional  
 数值采样方法，包括均值、最小和最大值，及和. The default is ["mean","min","max","sum"].  
 group\_column : string, optional  
 分组列名. The default is None.  
 geometry\_column : string, optional  
 几何列. The default is None.  
  
 Returns  
 -------  
 GeoDataFrame  
 重采样后时空数据.  
  
 '''   
 from pandas.api.types import is\_datetime64\_any\_dtype as is\_datetime  
 import pandas as pd  
 import geopandas as gpd  
 from tqdm import tqdm  
 from shapely import wkt  
   
 df\_notna=df[df[val\_column].notna()]  
 if is\_datetime(df\_notna[time\_column]):  
 df\_notna.rename(columns={time\_column:"ts"},inplace=True)  
 else:  
 df\_notn['ts']=pd.to\_datetime(df\_notna[time\_column])  
 df\_notna.sort\_values(by=["ts"],inplace=True)  
 df\_notna[val\_column]=df\_notna[val\_column].astype("float")  
 if geometry\_column:  
 crs=df\_notna.crs  
 print(f"CRS={crs}")  
   
 if group\_column:  
 df\_group=df\_notna.groupby(group\_column)   
 nodeID\_geometry\_mapping={}  
 g\_v\_resample\_lst=[]  
 for g\_n,g\_v in tqdm(df\_group):  
 g\_v.set\_index('ts',inplace=True)   
 if geometry\_column:  
 nodeID\_geometry\_mapping[g\_n]=g\_v.iloc[[0]].geometry.values[0].wkt  
 g\_v\_resample\_r\_dict={}  
 for r in rules:   
 g\_v\_resample=g\_v[val\_column].resample(r)  
 g\_v\_resample\_methods\_dict={}  
 for m in methods:  
 if m=="mean":  
 g\_v\_resample\_mean=g\_v\_resample.mean()  
 g\_v\_resample\_methods\_dict["mean"]=g\_v\_resample\_mean  
 elif m=="min":  
 g\_v\_resample\_min=g\_v\_resample.min()  
 g\_v\_resample\_methods\_dict["min"]=g\_v\_resample\_min=g\_v\_resample\_min  
 elif m=="max":  
 g\_v\_resample\_max=g\_v\_resample.max()  
 g\_v\_resample\_methods\_dict["max"]=g\_v\_resample\_min=g\_v\_resample\_max  
 elif m=="sum":  
 g\_v\_resample\_max=g\_v\_resample.sum()  
 g\_v\_resample\_methods\_dict["sum"]=g\_v\_resample\_min=g\_v\_resample\_max   
 else:  
 pass  
 g\_v\_resample\_r\_dict[r]=pd.concat(g\_v\_resample\_methods\_dict,axis=1)   
   
 g\_v\_resample\_r\_df=pd.concat(g\_v\_resample\_r\_dict,axis=1)   
 g\_v\_resample\_r\_df.columns=g\_v\_resample\_r\_df.columns.map("\_".join).str.strip("\_")  
 g\_v\_resample\_r\_df[group\_column]=g\_n  
 g\_v\_resample\_lst.append(g\_v\_resample\_r\_df)  
   
 g\_v\_resample\_df=pd.concat(g\_v\_resample\_lst)  
 if geometry\_column:  
 g\_v\_resample\_df[geometry\_column]=g\_v\_resample\_df[group\_column].map(nodeID\_geometry\_mapping)  
 g\_v\_resample\_df[geometry\_column]=g\_v\_resample\_df[geometry\_column].apply(wkt.loads)  
 g\_v\_resample\_df=gpd.GeoDataFrame(g\_v\_resample\_df,geometry=geometry\_column,crs=crs)   
 g\_v\_resample\_df.reset\_index(inplace=True)  
 return g\_v\_resample\_df  
   
 else:  
 df\_notna.set\_index('ts',inplace=True)   
 df\_notna\_resample\_r\_dict={}  
 for r in rules:   
 df\_notna\_resample=df\_notna[val\_column].resample(r)  
 df\_notna\_resample\_methods\_dict={}  
 for m in methods:  
 if m=="mean":  
 df\_notna\_resample\_mean=df\_notna\_resample.mean()  
 df\_notna\_resample\_methods\_dict["mean"]=df\_notna\_resample\_mean  
 elif m=="min":  
 df\_notna\_resample\_min=df\_notna\_resample.min()  
 df\_notna\_resample\_methods\_dict["min"]=df\_notna\_resample\_min  
 elif m=="max":  
 df\_notna\_resample\_max=df\_notna\_resample.max()  
 df\_notna\_resample\_methods\_dict["max"]=df\_notna\_resample\_max  
 elif m=="sum":  
 df\_notna\_resample\_max=df\_notna\_resample.sum()  
 df\_notna\_resample\_methods\_dict["sum"]=df\_notna\_resample\_max   
 else:  
 pass   
 df\_notna\_resample\_r\_dict[r]=pd.concat(df\_notna\_resample\_methods\_dict,axis=1)   
 df\_notna\_resample\_r\_df=pd.concat(df\_notna\_resample\_r\_dict,axis=1)   
 df\_notna\_resample\_r\_df.reset\_index(inplace=True)  
 return df\_notna\_resample\_r\_df

因为报告的犯罪记录为事件发生相关信息，每一条为一个事件，为按组按时间长度重采样计算各组给定时间长度下的事件数量，定义num列，并为各个样本赋值为1，表示每个样本为一件犯罪记录，用于df\_group\_resample()函数的val\_column参数。首先配置rules参数值为["M"]，按组按月计算犯罪发生数量。

crimes\_gdf['num']=1  
crimes\_M\_gdf=df\_group\_resample(crimes\_gdf.copy(deep=True),"num","ts",["M"],["sum"],group\_column="area\_numbe",geometry\_column="geometry")   
crimes\_M\_gdf

CRS=GEOGCS["WGS84(DD)",DATUM["WGS84",SPHEROID["WGS84",6378137,298.257223563]],PRIMEM["Greenwich",0],UNIT["degree",0.0174532925199433],AXIS["Longitude",EAST],AXIS["Latitude",NORTH]]  
  
  
100%|██████████| 77/77 [00:10<00:00, 7.20it/s]

|  | **ts** | **M\_sum** | **area\_numbe** | **geometry** |
| --- | --- | --- | --- | --- |
| **0** | 2001-01-31 | 11.0 | 1 | POLYGON ((-87.65456 41.99817, -87.65574 41.998... |
| **1** | 2001-02-28 | 3.0 | 1 | POLYGON ((-87.65456 41.99817, -87.65574 41.998... |
| **2** | 2001-03-31 | 4.0 | 1 | POLYGON ((-87.65456 41.99817, -87.65574 41.998... |
| **3** | 2001-04-30 | 5.0 | 1 | POLYGON ((-87.65456 41.99817, -87.65574 41.998... |
| **4** | 2001-05-31 | 3.0 | 1 | POLYGON ((-87.65456 41.99817, -87.65574 41.998... |
| **...** | ... | ... | ... | ... |
| **20231** | 2022-07-31 | 216.0 | 77 | POLYGON ((-87.65456 41.99817, -87.65456 41.998... |
| **20232** | 2022-08-31 | 268.0 | 77 | POLYGON ((-87.65456 41.99817, -87.65456 41.998... |
| **20233** | 2022-09-30 | 261.0 | 77 | POLYGON ((-87.65456 41.99817, -87.65456 41.998... |
| **20234** | 2022-10-31 | 283.0 | 77 | POLYGON ((-87.65456 41.99817, -87.65456 41.998... |
| **20235** | 2022-11-30 | 173.0 | 77 | POLYGON ((-87.65456 41.99817, -87.65456 41.998... |

20236 rows × 4 columns

数据量较大，重采样计算花费一定的时间，为避免重复计算，将计算结果存储为GPKG格式数据，并配置层为M，方便后续计算数据同样按层存储到该文件下。读取时按层读取，方便数据规整，避免冗余数据文件。

crimes\_M\_gdf.to\_file(args.data.crimes\_resample,driver='GPKG',layer='M')

crimes\_Y\_gdf=df\_group\_resample(crimes\_gdf.copy(deep=True),"num","ts",["Y"],["sum"],group\_column="area\_numbe",geometry\_column="geometry")   
crimes\_Y\_gdf.to\_file(args.data.crimes\_resample,driver='GPKG',layer='Y')

CRS=GEOGCS["WGS84(DD)",DATUM["WGS84",SPHEROID["WGS84",6378137,298.257223563]],PRIMEM["Greenwich",0],UNIT["degree",0.0174532925199433],AXIS["Longitude",EAST],AXIS["Latitude",NORTH]]  
  
  
100%|██████████| 77/77 [00:10<00:00, 7.52it/s]

### 2.7.4.2 空间马尔可夫链（Spatially Markov chain）[2]

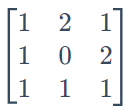
#### 1）经典离散（时间）马尔可夫链

马尔可夫链（Markov chain）又称离散时间马尔科夫链（Discrete-time Markov chain，D(T)MC），因俄国数学家安德烈·安德烈耶维奇·马尔可夫（俄语：Андре́й Андре́евич Ма́рков，英语：Andrey Andreyevich Markov）得名，为状态空间（ state space ）中经过从一个状态到另一个状态转换的随机过程。该过程要求具备“无记忆”的性质，为下一状态的概率分布只能由当前状态决定，在时间序列中它前面的事件均与之无关，该种特定类型的“无记忆性”称作马尔可夫性质（ Markov property）。在马尔可夫链的每一步，系统根据概率分布可以从一个状态变成另一个状态，也可以保持当前状态，状态的改变叫做转移（ transitions），与不同的状态改变相关的概率叫做转移概率（ transition probabilities）。

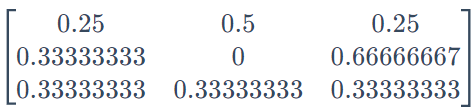
[PySAL-giddy](https://pysal.org/notebooks/explore/giddy/intro.html)⑯在阐释经典离散马尔可夫链时，给出了如下的示例，定义数据c为一个状态分布矩阵，包含有3个状态，分别为a，b和c（可理解为土地利用类型，例如林地、草地和建设用地；或股市，例如牛市、熊市和横盘；或人活动，例如吃、玩和睡，及天气情况、收入阶层等）；有5个样本（行）和3个时刻的状态分布（列），例如第一行样本初始状态为b，第2时刻状态为a，第3时刻状态为c，且后一状态仅与邻接的前一状态有关，以此类推。通过PySAL库提供的giddy.markov.Markov方法，给定状态分布矩阵，计算马尔可夫链的状态转移矩阵（transitions）、状态转移概率矩阵（ transition probabilities）和平稳状态分布（steady\_state distribution）等。马尔科夫链平稳状态为转移矩阵在转移概率矩阵作用下达到的平稳状态，以下述案例为例，如果在某一时刻，转移概率如何达到  
[0.307692310.288461540.40384615][0.30769231​0.28846154​0.40384615​]时，之后的任何分布都是同一分布。

import numpy as np  
c=np.array([['b','a','c'],['c','c','a'],['c','b','c'],['a','a','b'],['a','b','c']])  
c

array([['b', 'a', 'c'],  
 ['c', 'c', 'a'],  
 ['c', 'b', 'c'],  
 ['a', 'a', 'b'],  
 ['a', 'b', 'c']], dtype='<U1')

通过样本的状态分布数据，计算的转移矩阵为

，横纵坐标分别对应状态a、b和c，从中可以得知状态a到a的次数为1次，状态a到b的次数为2次，状态a到c的次数为1次，以状态a为开始状态的情景有4次；状态b到a、b、c的次数分别为1、0和2，以状态b为开始状态的情景有3次，依次类推，统计了状态分布数据中所有状态转移的数量。

而转移概率矩阵，计算结果为

** ni,jijkija bna,b=2** ，表明从状态a转移到状态b的概率为0.5，高于保持自身状态和转移至状态c的概率0.25。当样本处于a状态时，则下一状态更可能转移为状态b。

基于转移概率矩阵获得平稳状态分布，对应状态a、b和c概率为  
[0.30769231 ​0.28846154 ​0.40384615​]。

用giddy.ergodic.fmpt方法可以估计从一个状态（类别）到另一个状态的平均通过时间（步骤数），例如从状态a到b越需要2.200步，而到c需要2.573步，返回到自身a则需要3.25步。

import giddy  
m=giddy.markov.Markov(c)  
  
def markov\_info\_print(m,decimal=None):  
 '''  
 打印PySAL-giddy库giddy.markov.Markov方法计算离散时间马尔科夫链结果，包括分类名、转移矩阵、转移概率矩阵和平稳状态转移概率  
  
 Parameters  
 ----------  
 m : giddy.markov.Markov  
 giddy.markov.Markov方法计算离散时间马尔科夫链结果.  
 decimal : int, optional  
 转移概率矩阵和平稳状态概率小数位数. The default is None.  
  
 Returns  
 -------  
 None.  
  
 '''   
 import numpy as np  
 import giddy  
 import pandas as pd  
 np.set\_printoptions(suppress=True)  
   
 print('-'\*50)  
 m\_classes=m.classes   
 print(f"Unique classes: {m\_classes}")  
 df\_from\_nested\_lst=lambda lst,cols,idxes:pd.DataFrame(lst,columns=cols,index=idxes)   
 print(f"Transition matrix:\n {df\_from\_nested\_lst(m.transitions,m\_classes,m\_classes)}")  
 if decimal:  
 print(f"Transition probability matrix:\n {df\_from\_nested\_lst(m.p.round(decimal),m\_classes,m\_classes)}")  
 print(f"ergodic matrix:\n {df\_from\_nested\_lst(giddy.ergodic.fmpt(m.p).round(decimal),m\_classes,m\_classes)}")  
 print(f"Steady\_state: {m.steady\_state.round(decimal)}")  
   
 else:  
 print(f"Transition probability matrix:\n {df\_from\_nested\_lst(m.p,m\_classes,m\_classes)}")  
 print(f"ergodic matrix:\n {df\_from\_nested\_lst(giddy.ergodic.fmpt(m.p),m\_classes,m\_classes)}")  
 print(f"Steady\_state: {m.steady\_state}")   
   
markov\_info\_print(m,decimal=3)

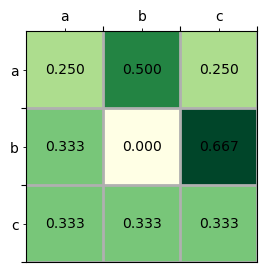
The Markov Chain is irreducible and is composed by:  
1 Recurrent class (indices):  
[0 1 2]  
0 Transient classes.  
The Markov Chain has 0 absorbing states.  
--------------------------------------------------  
Unique classes: ['a' 'b' 'c']  
Transition matrix:  
 a b c  
a 1.0 2.0 1.0  
b 1.0 0.0 2.0  
c 1.0 1.0 1.0  
Transition probability matrix:  
 a b c  
a 0.250 0.500 0.250  
b 0.333 0.000 0.667  
c 0.333 0.333 0.333  
ergodic matrix:  
 a b c  
a 3.25 2.200 2.571  
b 3.00 3.467 1.857  
c 3.00 2.600 2.476  
Steady\_state: [0.308 0.288 0.404]

sum(m.steady\_state)

1.0

直接打印转移概率矩阵并不方便观察数据变化关系，因此定义colorplot\_2d\_array\_with\_labels()函数，标注轴向标签，并根据值大小显示颜色变化。从图表可以观察到和转移概率值较高，而自身则为0。

def colorplot\_2d\_array\_with\_labels(array,labels,\*\*args):  
 '''  
 图表打印二维数组方阵，并标识轴向标签（分类名）  
  
 Parameters  
 ----------  
 array : array（2d）  
 二维数组.  
 labels : list  
 标签.  
 \*\*args : TYPE  
 配置图表打印样式参数，默认为：  
 setting=dict(figsize=(15,15),  
 cmap='YlGn',  
 xlabel\_rotation=0).  
  
 Returns  
 -------  
 None.  
  
 '''   
 import matplotlib.pyplot as plt  
 import matplotlib.cm as cm  
 import matplotlib.ticker as ticker  
   
 setting=dict(figsize=(15,15),  
 cmap='YlGn',  
 xlabel\_rotation=0)  
 setting.update(args)  
   
 nx, ny=array.shape  
 indx, indy=np.arange(nx), np.arange(ny)  
 x, y=np.meshgrid(indx, indy)   
   
 fig, ax=plt.subplots(figsize=setting['figsize'])  
 ax.imshow(array, interpolation="nearest", cmap=setting['cmap'])   
   
 for xval, yval in zip(x.flatten(), y.flatten()):  
 zval=array[yval, xval]  
 t="%.3f"%(zval)   
 ax.text(xval, yval, t,va='center', ha='center')   
   
 ax.set\_xticks(indx+0.5) # offset x/y ticks so gridlines run on border of boxes  
 ax.set\_yticks(indy+0.5)  
 ax.grid(ls='-', lw=2)  
   
 for a, ind, labels in zip((ax.xaxis, ax.yaxis), (indx, indy), (labels, labels)):  
 a.set\_major\_formatter(ticker.NullFormatter())  
 a.set\_minor\_locator(ticker.FixedLocator(ind))  
 a.set\_minor\_formatter(ticker.FixedFormatter(labels))  
   
 ax.xaxis.tick\_top()  
 ax.tick\_params(axis="x", which="both", rotation=setting['xlabel\_rotation'])  
 plt.show()   
  
colorplot\_2d\_array\_with\_labels(m.p.round(3),m.classes,figsize=(3,3))



**MCD12Q1\_v006-LCCS1土地覆盖类型变化的离散时间马尔科夫链计算**

用前文提取的MCD12Q1\_v006土地覆盖类型数据集中的粮农组织-土地覆盖分类系统1（LCCS1）土地覆盖层2001至2020历年变化采样点（去除空值，有效采样点为3233条）数据，计算离散时间马尔科夫链，获取转移概率矩阵。

import pickle  
  
with open(args.data.MCD12Q1v006\_years\_vals\_array,'rb') as f:  
 MCD12Q1v006\_years\_vals=pickle.load(f)  
MCD12Q1v006\_years\_vals.shape

(3586, 20)

移除空值。

MCD12Q1v006\_years\_vals\_notna=MCD12Q1v006\_years\_vals[~np.isnan(MCD12Q1v006\_years\_vals).any(axis=1), :]  
print(MCD12Q1v006\_years\_vals\_notna.shape)  
MCD12Q1v006\_years\_vals\_notna

(3233, 20)  
  
  
  
  
  
array([[31., 31., 31., ..., 31., 31., 31.],  
 [ 1., 1., 1., ..., 1., 1., 1.],  
 [ 1., 1., 1., ..., 1., 1., 1.],  
 ...,  
 [31., 31., 31., ..., 31., 31., 31.],  
 [ 1., 1., 1., ..., 1., 1., 1.],  
 [22., 22., 22., ..., 22., 22., 22.]], dtype=float32)

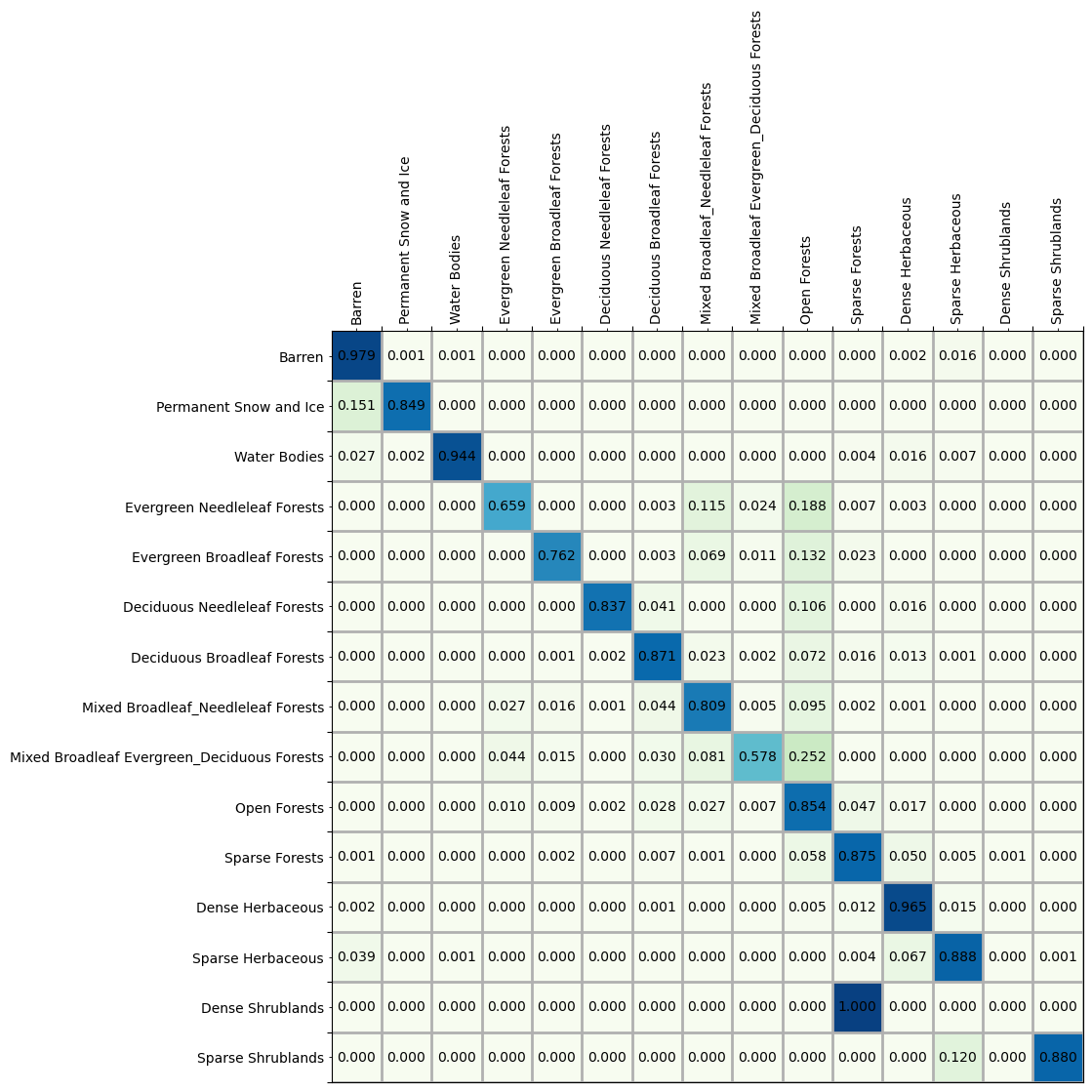
计算计算离散时间马尔科夫链，并打印结果。

import giddy  
MCD12Q1v006\_years\_vals\_markov=giddy.markov.Markov(MCD12Q1v006\_years\_vals\_notna)  
markov\_info\_print(MCD12Q1v006\_years\_vals\_markov,decimal=3)

The Markov Chain is irreducible and is composed by:  
1 Recurrent class (indices):  
[ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14]  
0 Transient classes.  
The Markov Chain has 0 absorbing states.  
--------------------------------------------------  
Unique classes: [ 1. 2. 3. 11. 12. 13. 14. 15. 16. 21. 22. 31. 32. 41. 43.]  
Transition matrix:  
 1.0 2.0 3.0 11.0 12.0 13.0 14.0 15.0 16.0 \  
1.0 16521.0 24.0 20.0 0.0 0.0 0.0 0.0 0.0 0.0   
2.0 22.0 124.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0   
3.0 15.0 1.0 522.0 0.0 0.0 0.0 0.0 0.0 0.0   
11.0 0.0 0.0 0.0 189.0 0.0 0.0 1.0 33.0 7.0   
12.0 0.0 0.0 0.0 0.0 266.0 0.0 1.0 24.0 4.0   
13.0 0.0 0.0 0.0 0.0 0.0 103.0 5.0 0.0 0.0   
14.0 0.0 0.0 0.0 0.0 2.0 4.0 1971.0 52.0 4.0   
15.0 0.0 0.0 0.0 38.0 22.0 2.0 61.0 1127.0 7.0   
16.0 0.0 0.0 0.0 6.0 2.0 0.0 4.0 11.0 78.0   
21.0 1.0 0.0 0.0 56.0 50.0 11.0 164.0 157.0 38.0   
22.0 6.0 0.0 2.0 1.0 9.0 0.0 36.0 5.0 0.0   
31.0 35.0 0.0 9.0 2.0 0.0 2.0 34.0 1.0 0.0   
32.0 219.0 1.0 4.0 0.0 0.0 0.0 2.0 0.0 0.0   
41.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0   
43.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0   
  
 21.0 22.0 31.0 32.0 41.0 43.0   
1.0 1.0 7.0 33.0 262.0 0.0 1.0   
2.0 0.0 0.0 0.0 0.0 0.0 0.0   
3.0 0.0 2.0 9.0 4.0 0.0 0.0   
11.0 54.0 2.0 1.0 0.0 0.0 0.0   
12.0 46.0 8.0 0.0 0.0 0.0 0.0   
13.0 13.0 0.0 2.0 0.0 0.0 0.0   
14.0 162.0 36.0 29.0 2.0 0.0 0.0   
15.0 132.0 3.0 1.0 0.0 0.0 0.0   
16.0 34.0 0.0 0.0 0.0 0.0 0.0   
21.0 4952.0 272.0 96.0 0.0 0.0 0.0   
22.0 289.0 4385.0 249.0 23.0 4.0 0.0   
31.0 106.0 275.0 22020.0 342.0 1.0 0.0   
32.0 0.0 24.0 376.0 5017.0 0.0 4.0   
41.0 0.0 5.0 0.0 0.0 0.0 0.0   
43.0 0.0 0.0 0.0 3.0 0.0 22.0   
Transition probability matrix:  
 1.0 2.0 3.0 11.0 12.0 13.0 14.0 15.0 16.0 21.0 \  
1.0 0.979 0.001 0.001 0.000 0.000 0.000 0.000 0.000 0.000 0.000   
2.0 0.151 0.849 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000   
3.0 0.027 0.002 0.944 0.000 0.000 0.000 0.000 0.000 0.000 0.000   
11.0 0.000 0.000 0.000 0.659 0.000 0.000 0.003 0.115 0.024 0.188   
12.0 0.000 0.000 0.000 0.000 0.762 0.000 0.003 0.069 0.011 0.132   
13.0 0.000 0.000 0.000 0.000 0.000 0.837 0.041 0.000 0.000 0.106   
14.0 0.000 0.000 0.000 0.000 0.001 0.002 0.871 0.023 0.002 0.072   
15.0 0.000 0.000 0.000 0.027 0.016 0.001 0.044 0.809 0.005 0.095   
16.0 0.000 0.000 0.000 0.044 0.015 0.000 0.030 0.081 0.578 0.252   
21.0 0.000 0.000 0.000 0.010 0.009 0.002 0.028 0.027 0.007 0.854   
22.0 0.001 0.000 0.000 0.000 0.002 0.000 0.007 0.001 0.000 0.058   
31.0 0.002 0.000 0.000 0.000 0.000 0.000 0.001 0.000 0.000 0.005   
32.0 0.039 0.000 0.001 0.000 0.000 0.000 0.000 0.000 0.000 0.000   
41.0 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000   
43.0 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000   
  
 22.0 31.0 32.0 41.0 43.0   
1.0 0.000 0.002 0.016 0.000 0.000   
2.0 0.000 0.000 0.000 0.000 0.000   
3.0 0.004 0.016 0.007 0.000 0.000   
11.0 0.007 0.003 0.000 0.000 0.000   
12.0 0.023 0.000 0.000 0.000 0.000   
13.0 0.000 0.016 0.000 0.000 0.000   
14.0 0.016 0.013 0.001 0.000 0.000   
15.0 0.002 0.001 0.000 0.000 0.000   
16.0 0.000 0.000 0.000 0.000 0.000   
21.0 0.047 0.017 0.000 0.000 0.000   
22.0 0.875 0.050 0.005 0.001 0.000   
31.0 0.012 0.965 0.015 0.000 0.000   
32.0 0.004 0.067 0.888 0.000 0.001   
41.0 1.000 0.000 0.000 0.000 0.000   
43.0 0.000 0.000 0.120 0.000 0.880   
ergodic matrix:  
 1.0 2.0 3.0 11.0 12.0 13.0 14.0 \  
1.0 4.347 2772.881 1857.875 743.033 834.375 2971.143 349.698   
2.0 6.636 418.831 1864.511 749.669 841.011 2977.780 356.334   
3.0 126.676 2802.282 107.608 720.948 812.100 2949.352 327.911   
11.0 285.172 3042.128 1996.280 170.795 606.619 2765.936 144.770   
12.0 285.304 3042.255 1996.243 510.216 146.017 2771.662 150.402   
13.0 283.222 3040.183 1994.684 545.729 641.192 452.275 131.899   
14.0 282.941 3039.899 1994.391 540.264 635.269 2754.104 22.278   
15.0 287.085 3044.042 1998.240 455.117 574.645 2749.518 121.976   
16.0 284.856 3041.813 1995.978 456.728 589.801 2764.786 137.414   
21.0 280.307 3037.265 1991.393 520.904 617.110 2766.404 152.543   
22.0 265.592 3022.481 1974.876 593.415 679.738 2830.340 205.037   
31.0 241.369 2998.365 1955.309 659.257 751.027 2886.517 265.960   
32.0 165.956 2923.570 1918.533 695.273 786.691 2923.167 301.683   
41.0 266.592 3023.481 1975.876 594.415 680.738 2831.340 206.037   
43.0 174.290 2931.904 1926.866 703.607 795.024 2931.500 310.017   
  
 15.0 16.0 21.0 22.0 31.0 32.0 41.0 \  
1.0 411.436 1100.023 230.701 189.598 94.184 72.049 11448.297   
2.0 418.072 1106.660 237.337 196.234 100.820 78.685 11454.934   
3.0 389.343 1077.913 208.565 165.861 74.896 93.408 11425.395   
11.0 129.758 810.096 13.715 59.485 55.451 148.638 11335.326   
12.0 145.028 837.128 18.428 55.617 55.933 148.825 11332.274   
13.0 210.497 899.217 30.274 67.785 52.378 146.427 11341.493   
14.0 187.880 890.824 37.664 62.815 53.216 146.065 11337.612   
15.0 35.068 849.966 19.319 61.974 57.409 150.502 11337.715   
16.0 148.640 365.417 10.830 59.115 55.195 148.326 11334.978   
21.0 191.687 874.957 9.213 53.925 50.652 143.833 11329.909   
22.0 261.041 949.321 79.480 11.092 38.948 129.468 11283.904   
31.0 327.809 1016.391 147.076 108.798 2.635 102.697 11364.987   
32.0 363.669 1052.287 183.001 142.325 44.326 11.320 11400.540   
41.0 262.041 950.321 80.480 1.000 39.948 130.468 11284.904   
43.0 372.003 1060.621 191.335 150.659 52.660 8.333 11408.873   
  
 43.0   
1.0 13142.420   
2.0 13149.056   
3.0 13182.625   
11.0 13256.336   
12.0 13256.513   
13.0 13254.172   
14.0 13253.824   
15.0 13258.209   
16.0 13256.023   
21.0 13251.520   
22.0 13237.093   
31.0 13210.778   
32.0 13112.963   
41.0 13238.093   
43.0 1574.556   
Steady\_state: [0.23 0.002 0.009 0.006 0.007 0.002 0.045 0.029 0.003 0.109 0.09 0.379  
 0.088 0. 0.001]

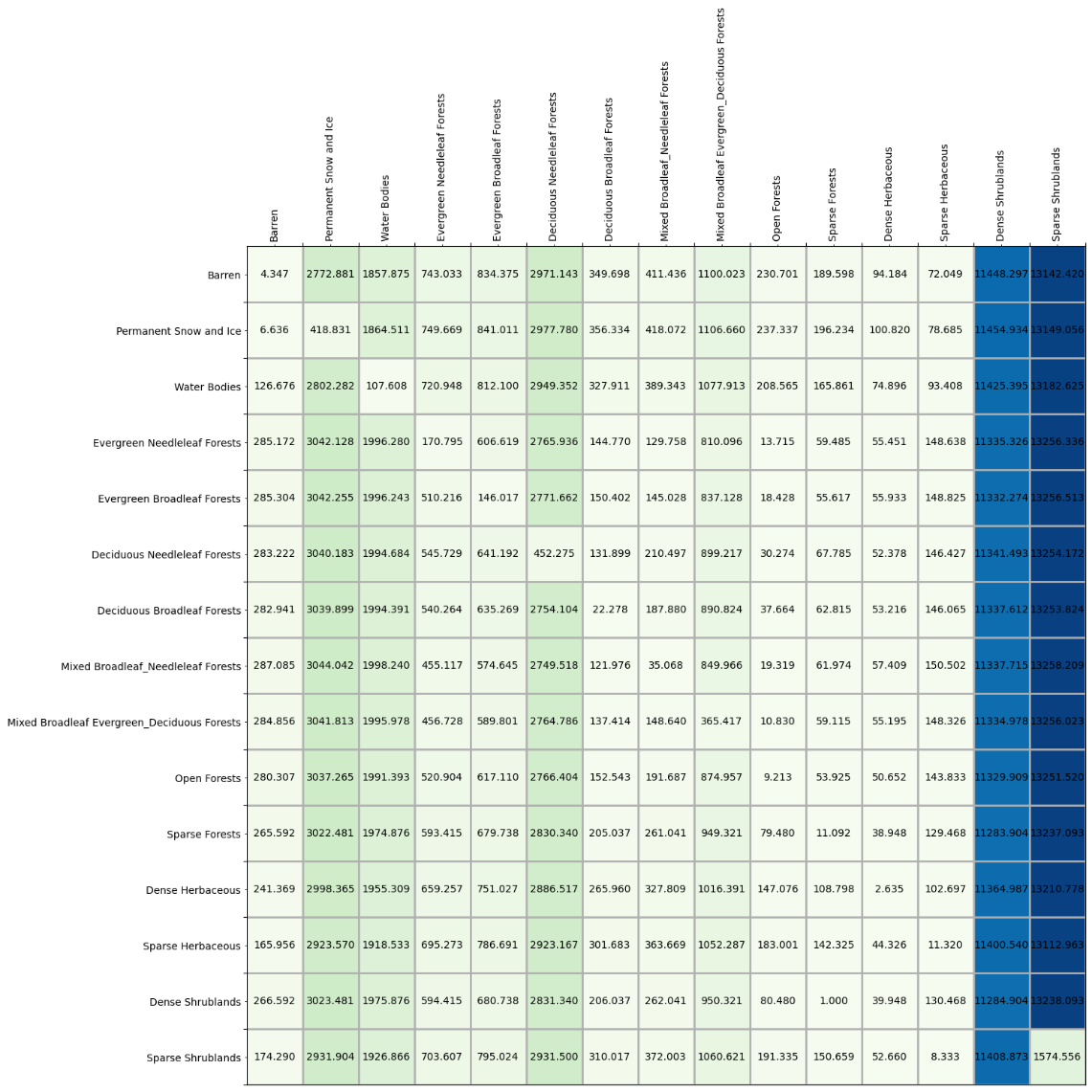
对于分类数量较多的二维转移概率矩阵，打印图表更易于观察不同分类间的转移程度。从下述图表可以观察到，这20年间，除了’Dense Shrublands’全部转移为’Sparse Forests’外，各类别通常保持不变，但是有多数不同类的Forests会小部分的转移为’Open Forests’类型，及更少更小转移概率的到’Mixed Broadleaf\_Needleleaf Forests’类型；同时可以注意到，‘Permanent Snow and Ice’有0.151的概率转换为’Barren’。

LCCS1\_mapping={1:'Barren',  
 2:'Permanent Snow and Ice',  
 3:'Water Bodies',  
 11:'Evergreen Needleleaf Forests',  
 12:'Evergreen Broadleaf Forests',  
 13:'Deciduous Needleleaf Forests',  
 14:'Deciduous Broadleaf Forests',  
 15:'Mixed Broadleaf\_Needleleaf Forests',  
 16:'Mixed Broadleaf Evergreen\_Deciduous Forests',  
 21:'Open Forests',  
 22:'Sparse Forests',  
 31:'Dense Herbaceous',  
 32:'Sparse Herbaceous',  
 41:'Dense Shrublands',  
 42:'Shrubland\_Grassland Mosaics',  
 43:'Sparse Shrublands',  
 255:'Unclassiﬁed'  
 }  
  
LCCS1\_labels=[LCCS1\_mapping[i] for i in MCD12Q1v006\_years\_vals\_markov.classes]  
colorplot\_2d\_array\_with\_labels(MCD12Q1v006\_years\_vals\_markov.p.round(3),LCCS1\_labels,figsize=(10,10),xlabel\_rotation=90,cmap='GnBu')



打印土地覆盖类型转化时长（单位年，1步为1年），所有类型要转化为’Dense Shrublands’或’Sparse Shrublands’类型均需要万年时间；但也存在十几年甚至几年就能发生转化的类型，例如为6.636年，为8.333年等。

MCD12Q1v006\_years\_vals\_ergodic=giddy.ergodic.fmpt(MCD12Q1v006\_years\_vals\_markov.p).round(3)  
colorplot\_2d\_array\_with\_labels(MCD12Q1v006\_years\_vals\_ergodic,LCCS1\_labels,figsize=(15,15),xlabel\_rotation=90,cmap='GnBu')



计算各类转化的平均时间，可以得知各土地覆盖类型转化为’Dense Herbaceous’的平均时长为55.2434年，其次为’Sparse Forests’等。

MCD12Q1v006\_years\_vals\_ergodic\_mean=list(zip(MCD12Q1v006\_years\_vals\_ergodic.mean(axis=0),LCCS1\_labels))  
MCD12Q1v006\_years\_vals\_ergodic\_mean.sort(key=lambda x:x[0])  
MCD12Q1v006\_years\_vals\_ergodic\_mean

[(55.2434, 'Dense Herbaceous'),  
 (92.41886666666666, 'Sparse Forests'),  
 (99.82786666666668, 'Open Forests'),  
 (110.60293333333335, 'Sparse Herbaceous'),  
 (212.26393333333328, 'Deciduous Broadleaf Forests'),  
 (216.02300000000005, 'Barren'),  
 (256.9314666666667, 'Mixed Broadleaf\_Needleleaf Forests'),  
 (577.2913333333332, 'Evergreen Needleleaf Forests'),  
 (666.0904666666665, 'Evergreen Broadleaf Forests'),  
 (922.7428, 'Mixed Broadleaf Evergreen\_Deciduous Forests'),  
 (1836.5775333333333, 'Water Bodies'),  
 (2688.3882666666664, 'Deciduous Needleleaf Forests'),  
 (2812.092, 'Permanent Snow and Ice'),  
 (11361.4094, 'Dense Shrublands'),  
 (12442.278733333334, 'Sparse Shrublands')]

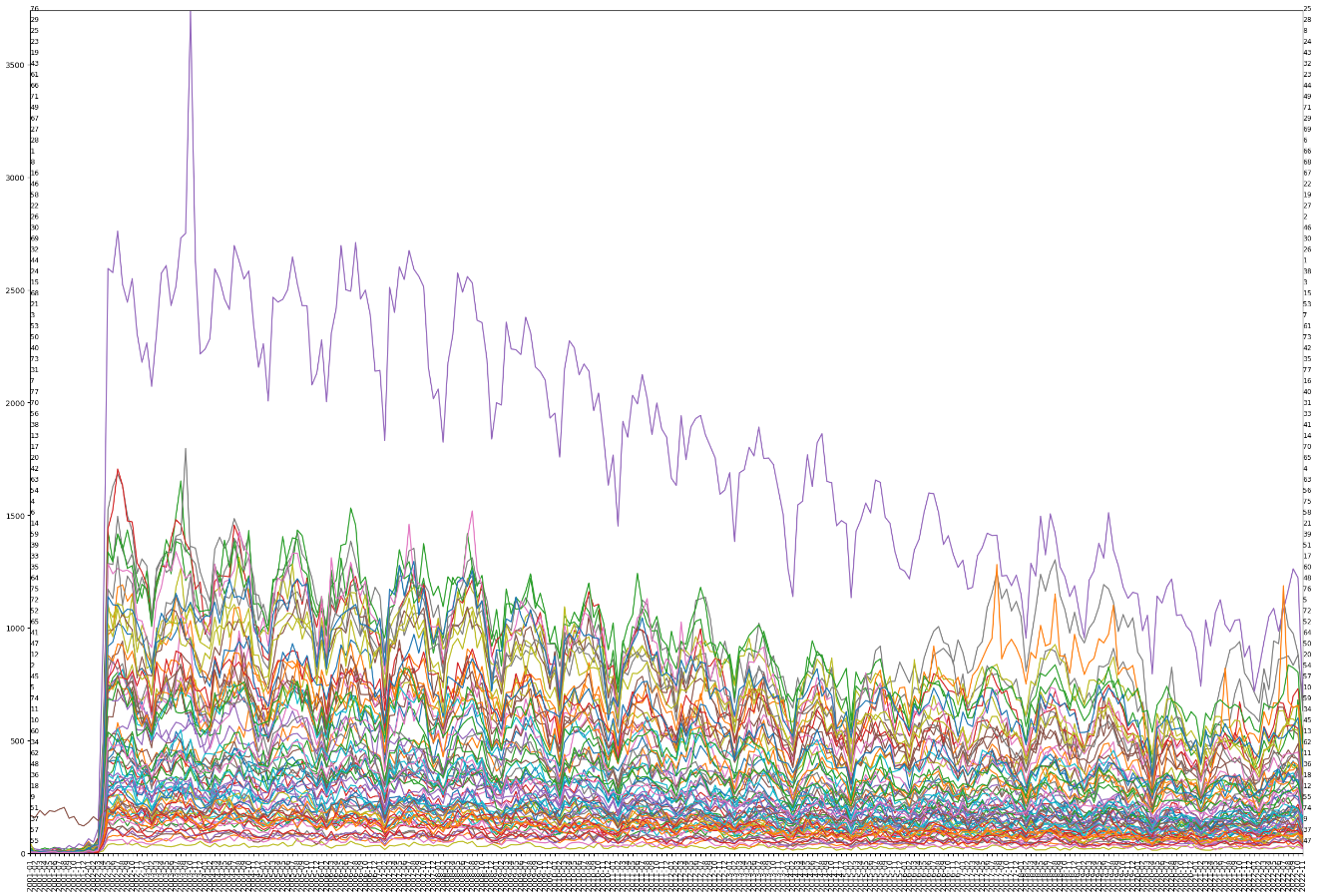
#### 2）时空数据绝对和相对动态变化图表打印

通过图表打印各社区历年数值，有绝对和相对动态两种变化曲线方便观察数据变化，定义函数为absoluteORrelative\_dynamics\_ts\_plot()。绝对曲线并不对数值做出任何改变，可以从图表中观察到各社区犯罪报告数量的绝对变化趋势；相对曲线则是各社区时间轴向上数值除以该社区所有数值的均值，方便观察时间轴向上各社区数值变化的相对趋势，是各社区数值趋于聚拢还是趋于扩散，如果都向均值聚拢，可以说明历年犯罪报告事件浮动较小；如何趋于扩散，则表明历年犯罪报告事件浮动较大。

下述代码打印了月和年采样的时间轴向数值绝对变化曲线。从月变化曲线可以得知每年各月份犯罪报告数以每年年中6、7、8月份居高，依次向两侧递减，至1、2月份降到最低；从年绝对变化曲线可以观察到各社区的犯罪报告数均趋于下降趋势，在2019年有一个明显的下滑，这与2019年末开始爆发的疫情可能有关。

import geopandas as gpd  
crimes\_M\_gdf=gpd.read\_file(args.data.crimes\_resample,layer='M')

def absoluteORrelative\_dynamics\_ts\_plot(df\_,datetime\_column,val\_column,group\_column,strftime\_pattern='%Y-%m',freq='M',fill\_val=-1,absoluteORrelative='absolute',\*\*setting):  
 '''  
 时间轴向面板数据的绝对变化和相对变化曲线。绝对变化保持数值不变，相对变化为时间轴向上各行数值除以对应行所有数值的均值  
  
 Parameters  
 ----------  
 df\_ : DataFrame  
 时间轴向面板数据.  
 datetime\_column : string  
 时间列名.  
 val\_column : string  
 值列名.  
 group\_column : string  
 分组列名.  
 strftime\_pattern : string, optional  
 时间格式化字符. The default is '%Y-%m'.  
 freq : string, optional  
 时间序列频率的别名. The default is 'M'.  
 fill\_val : numerical, optional  
 数值，浮点型或整型. The default is -1.  
 absoluteORrelative : string, optional  
 绝对变化或相对变化曲线，为absolute和relative. The default is 'absolute'.  
 \*\*setting : TYPE  
 图表打印样式，默认为：  
 setting\_dict=dict(figsize=(15,10),  
 rotation=90,  
 text\_fontsize=9,  
 ).  
  
 Returns  
 -------  
 val\_array : array  
 面板数组，如果为相对变化则为各行数值除以该行均值.  
 ts\_unique : Index  
 时间轴向时间索引.  
 group\_unique : Index  
 分组值.  
  
 '''   
 import pandas as pd  
 import numpy as np  
 import matplotlib.pyplot as plt  
 from pylab import rcParams  
   
 setting\_dict=dict(figsize=(15,10),  
 rotation=90,  
 text\_fontsize=9,  
 )  
 setting\_dict.update(setting)  
   
 df=df\_.copy(deep=True)  
 df['ts\_sft']=df[datetime\_column].apply(lambda x:x.strftime(strftime\_pattern))  
 df\_hierarchicalIdx=df.set\_index([group\_column,'ts\_sft']).sort\_index()  
 dt\_complement=pd.date\_range(df['ts\_sft'].min(),df['ts\_sft'].max(),freq=freq,inclusive="both").strftime(strftime\_pattern)  
 group\_unique=df\_hierarchicalIdx.index.get\_level\_values(group\_column).unique()   
 idx\_all=[(g,t) for g in group\_unique for t in dt\_complement]  
 idx\_missing=set(idx\_all)-set(df\_hierarchicalIdx.index)  
 for idx in idx\_missing:  
 df\_hierarchicalIdx.loc[idx,:]=None  
 df\_hierarchicalIdx.sort\_index(inplace=True)  
 df\_hierarchicalIdx[val\_column].fillna(value=fill\_val,inplace=True)  
 ts\_unique=df\_hierarchicalIdx.index.get\_level\_values('ts\_sft').unique()   
 val\_array\_lst=df\_hierarchicalIdx[val\_column].groupby(level=group\_column).apply(pd.Series.to\_numpy).to\_numpy()  
 val\_array=np.stack(val\_array\_lst).T  
   
 if absoluteORrelative=='absolute':  
 order\_ts\_0=np.argsort(val\_array[0])   
 order\_ts\_tailender=np.argsort(val\_array[-1])  
 val\_0=group\_unique[order\_ts\_0[::-1]]  
 val\_tailender=group\_unique[order\_ts\_tailender[::-1]]  
 first\_last=np.vstack((val\_0,val\_tailender))  
 elif absoluteORrelative=='relative':  
 val\_array=(val\_array.T / val\_array.mean(axis=1)).T  
 order\_ts\_0=np.argsort(val\_array[0])   
 order\_ts\_tailender=np.argsort(val\_array[-1])  
 val\_0=group\_unique[order\_ts\_0[::-1]]  
 val\_tailender=group\_unique[order\_ts\_tailender[::-1]]  
 first\_last=np.vstack((val\_0,val\_tailender))   
   
 plt.figure()  
 rcParams['figure.figsize']=setting\_dict['figsize']  
 plt.plot(ts\_unique,val\_array)  
   
 x\_min=min(ts\_unique)  
 x\_max=max(ts\_unique)  
 y\_min=np.min(val\_array)  
 y\_max=np.max(val\_array)  
 y\_unit=(y\_max-y\_min)/len(group\_unique)  
  
 for i in range(len(group\_unique)):  
 plt.text(x\_min,y\_max-(i\*y\_unit), first\_last[0][i],fontsize=setting\_dict['text\_fontsize'])  
 plt.text(x\_max,y\_max-(i\*y\_unit), first\_last[1][i],fontsize=setting\_dict['text\_fontsize'])  
   
 plt.xlim((x\_min, x\_max))  
 plt.ylim((y\_min, y\_max))  
 plt.xticks(rotation=setting\_dict['rotation'])  
 plt.show()  
   
 return val\_array,ts\_unique,group\_unique  
   
val\_array\_M,\_,\_=absoluteORrelative\_dynamics\_ts\_plot(crimes\_M\_gdf,'ts','M\_sum','area\_numbe',absoluteORrelative='absolute',figsize=(15\*2,10\*2));

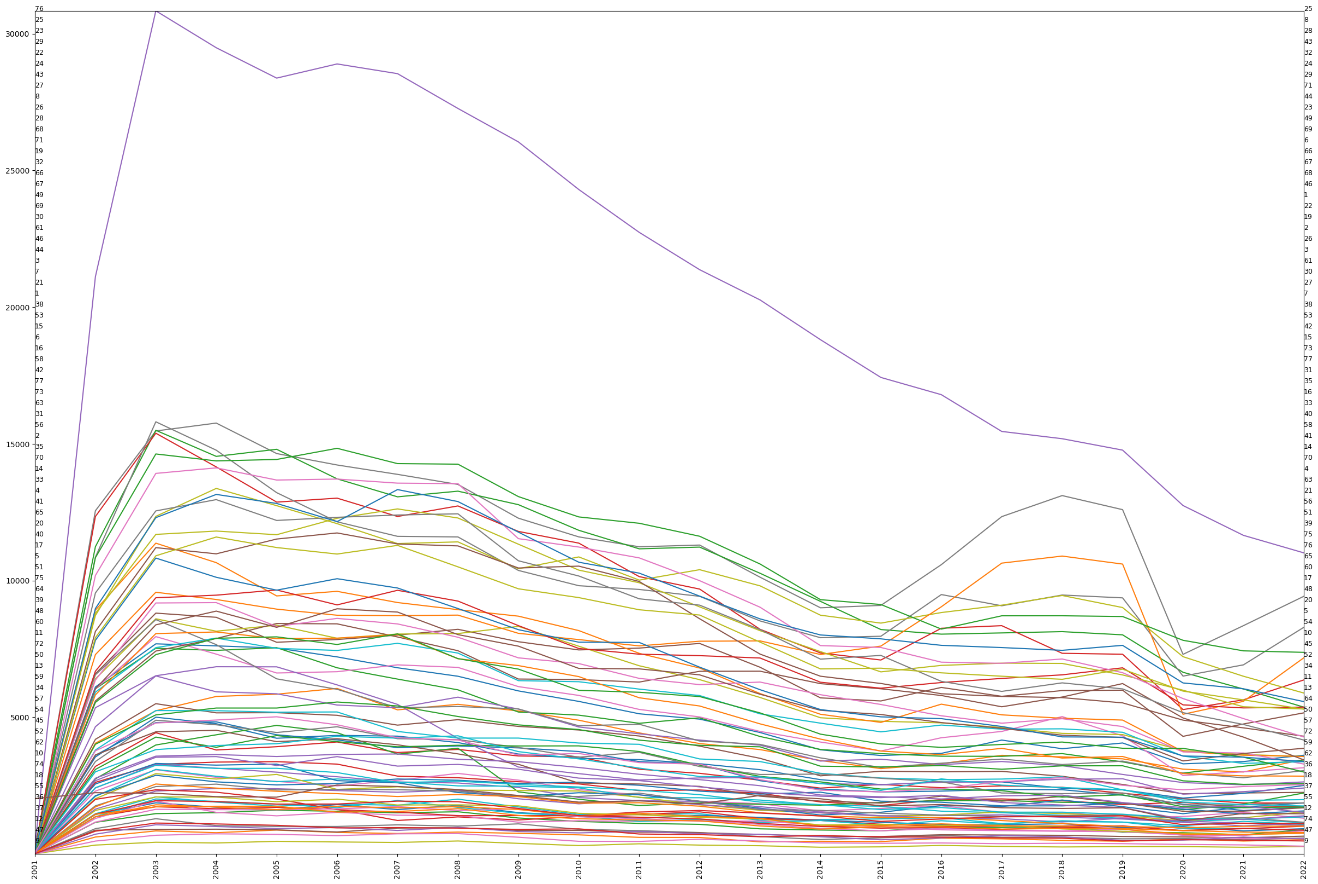


crimes\_Y\_gdf=gpd.read\_file(args.data.crimes\_resample,layer='Y')  
crimes\_Y\_gdf

|  | **ts** | **Y\_sum** | **area\_numbe** | **geometry** |
| --- | --- | --- | --- | --- |
| **0** | 2001-12-31 | 58.0 | 1 | POLYGON ((-87.65456 41.99817, -87.65574 41.998... |
| **1** | 2002-12-31 | 6106.0 | 1 | POLYGON ((-87.65456 41.99817, -87.65574 41.998... |
| **2** | 2003-12-31 | 7685.0 | 1 | POLYGON ((-87.65456 41.99817, -87.65574 41.998... |
| **3** | 2004-12-31 | 7610.0 | 1 | POLYGON ((-87.65456 41.99817, -87.65574 41.998... |
| **4** | 2005-12-31 | 7531.0 | 1 | POLYGON ((-87.65456 41.99817, -87.65574 41.998... |
| **...** | ... | ... | ... | ... |
| **1689** | 2018-12-31 | 2793.0 | 77 | POLYGON ((-87.65456 41.99817, -87.65456 41.998... |
| **1690** | 2019-12-31 | 2510.0 | 77 | POLYGON ((-87.65456 41.99817, -87.65456 41.998... |
| **1691** | 2020-12-31 | 2345.0 | 77 | POLYGON ((-87.65456 41.99817, -87.65456 41.998... |
| **1692** | 2021-12-31 | 2465.0 | 77 | POLYGON ((-87.65456 41.99817, -87.65456 41.998... |
| **1693** | 2022-12-31 | 2534.0 | 77 | POLYGON ((-87.65456 41.99817, -87.65456 41.998... |

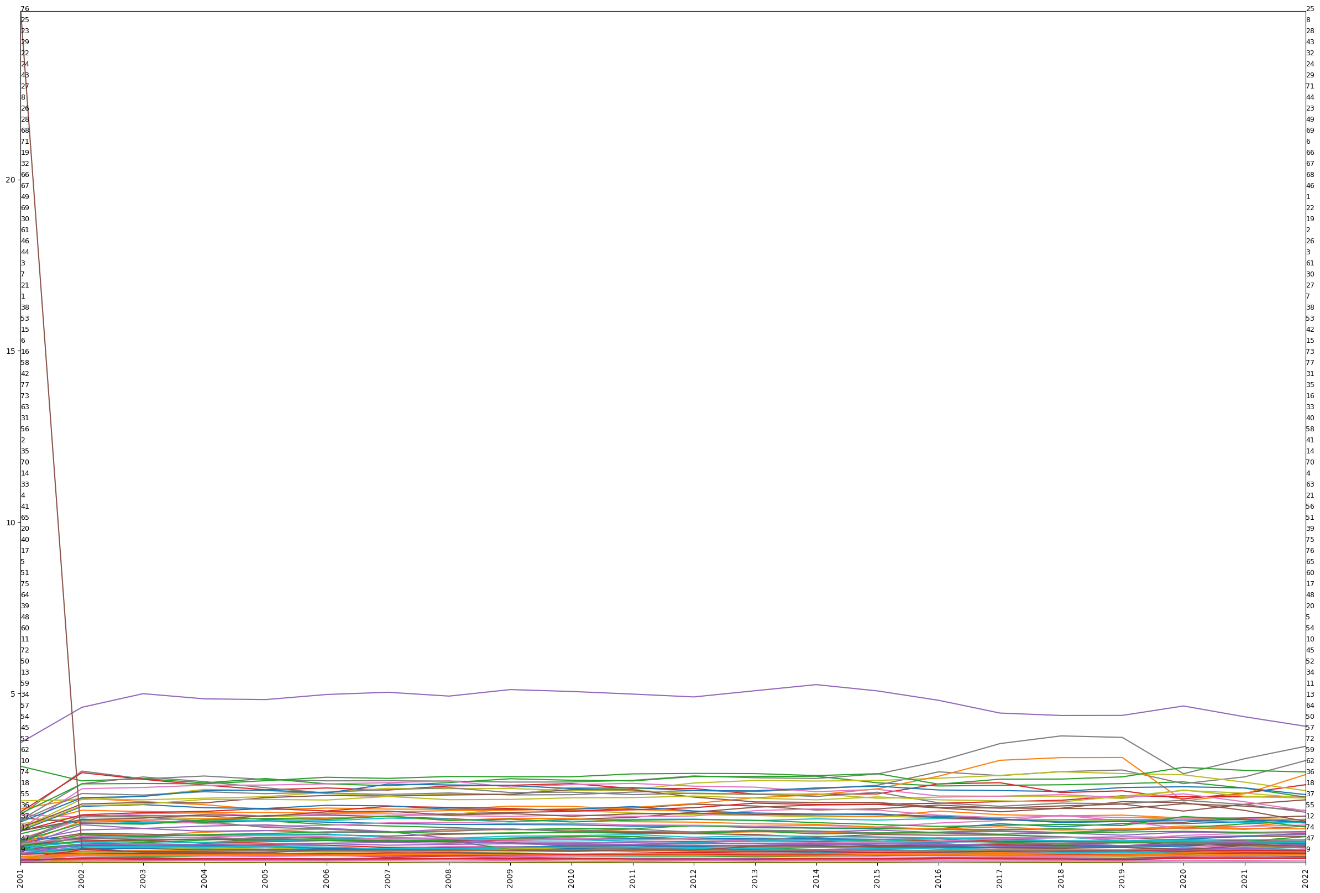
1694 rows × 4 columns

val\_array\_Y,ts\_unique,group\_unique=absoluteORrelative\_dynamics\_ts\_plot(crimes\_Y\_gdf,'ts','Y\_sum','area\_numbe',strftime\_pattern='%Y',freq='Y',absoluteORrelative='absolute',figsize=(15\*2,10\*2))



从相对变化曲线可以初步判断历年各社区的犯罪报告数的分布除个别社区基本保持不变。

absoluteORrelative\_dynamics\_ts\_plot(crimes\_Y\_gdf,'ts','Y\_sum','area\_numbe',strftime\_pattern='%Y',freq='Y',absoluteORrelative='relative',figsize=(15\*2,10\*2))



(array([[ 0.71444569, 0.51735722, 0.89921613, ..., 0.30795073,  
 24.9193729 , 0.5912654 ],  
 [ 1.32756746, 0.88033421, 1.31321773, ..., 0.60181899,  
 0.48702115, 0.8198914 ],  
 [ 1.24336551, 0.84713882, 1.21715533, ..., 0.5785654 ,  
 0.35885292, 0.77465635],  
 ...,  
 [ 1.19711332, 1.12663078, 1.07431384, ..., 0.61508736,  
 0.44106087, 0.85196684],  
 [ 1.25068399, 1.10962105, 1.18737228, ..., 0.64681614,  
 0.54796101, 0.91265128],  
 [ 1.32311852, 1.2659422 , 1.25897842, ..., 0.55673602,  
 0.53877679, 0.92874857]]),  
 Index(['2001', '2002', '2003', '2004', '2005', '2006', '2007', '2008', '2009',  
 '2010', '2011', '2012', '2013', '2014', '2015', '2016', '2017', '2018',  
 '2019', '2020', '2021', '2022'],  
 dtype='object', name='ts\_sft'),  
 Int64Index([ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,  
 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,  
 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51,  
 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68,  
 69, 70, 71, 72, 73, 74, 75, 76, 77],  
 dtype='int64', name='area\_numbe'))

#### 3）犯罪率年统计量的经典离散型马尔可夫链计算

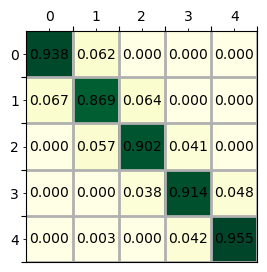
使用mapclassify库提供的Quantiles方法，按分位数将数值按时间点将各社区犯罪报告数划分为5个区间（分类过程），然后计算各社区5分位数时间轴向的变化情况。使用markov\_info\_print()函数打印计算结果，包括分类名、转移矩阵、转移概率矩阵和平稳状态转移概率等。使用colorplot\_2d\_array\_with\_labels()打印转移概率矩阵，可以观察到各类（数值的5分位数）基本保持不变，但是存在向所在分位数邻接分位数转移概率，离所在分位数越远，转换概率越低。

import mapclassify as mc  
q5=np.array([mc.Quantiles(y,k=5).yb for y in val\_array\_Y]).transpose()  
print(q5.shape)  
q5

(77, 22)  
  
  
  
  
  
array([[3, 3, 3, ..., 3, 3, 3],  
 [2, 2, 2, ..., 3, 3, 3],  
 [3, 3, 3, ..., 3, 3, 3],  
 ...,  
 [1, 2, 2, ..., 2, 2, 1],  
 [4, 1, 1, ..., 1, 1, 1],  
 [2, 2, 2, ..., 2, 2, 2]])

m5=giddy.markov.Markov(q5)  
markov\_info\_print(m5,decimal=3)  
colorplot\_2d\_array\_with\_labels(m5.p.round(3),m5.classes,figsize=(3,3))

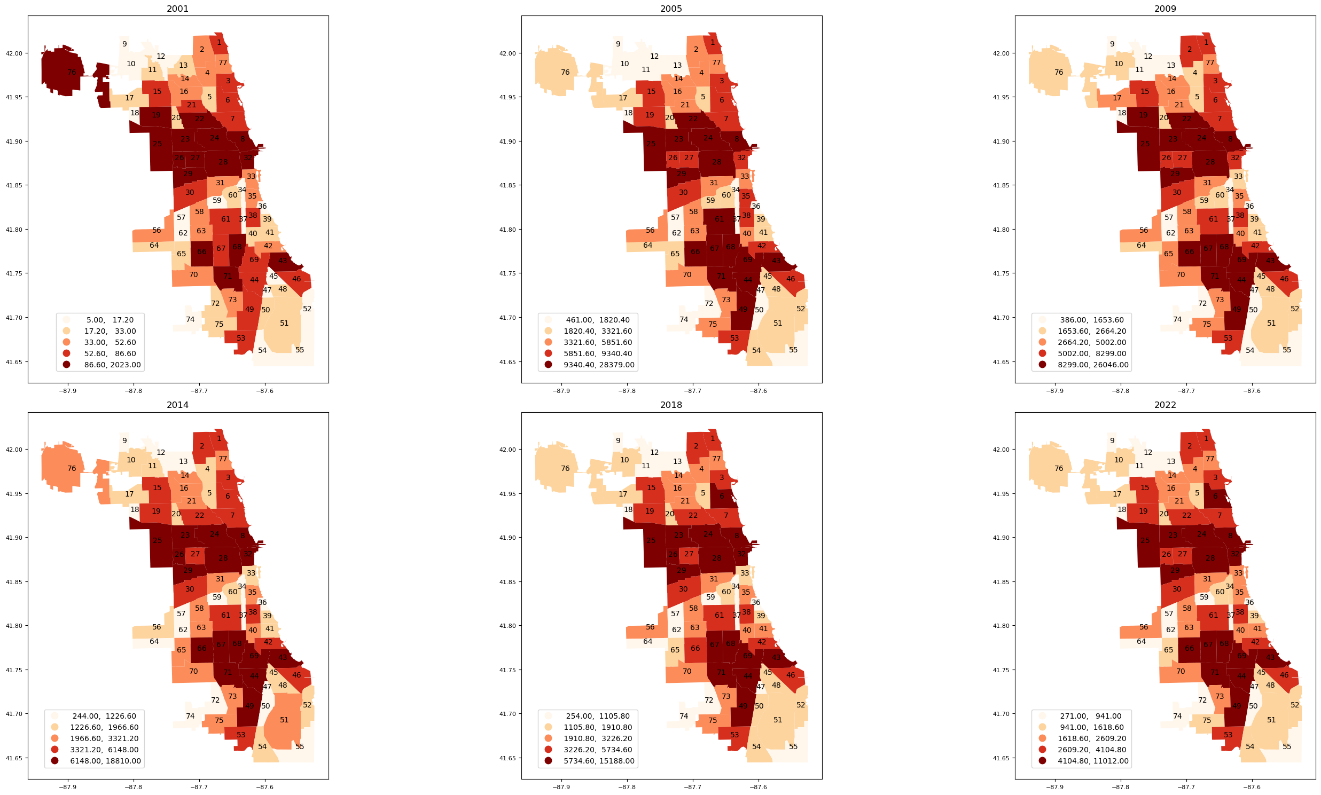
The Markov Chain is irreducible and is composed by:  
1 Recurrent class (indices):  
[0 1 2 3 4]  
0 Transient classes.  
The Markov Chain has 0 absorbing states.  
--------------------------------------------------  
Unique classes: [0 1 2 3 4]  
Transition matrix:  
 0 1 2 3 4  
0 317.0 21.0 0.0 0.0 0.0  
1 21.0 273.0 20.0 0.0 0.0  
2 0.0 18.0 284.0 13.0 0.0  
3 0.0 0.0 12.0 287.0 15.0  
4 0.0 1.0 0.0 14.0 321.0  
Transition probability matrix:  
 0 1 2 3 4  
0 0.938 0.062 0.000 0.000 0.000  
1 0.067 0.869 0.064 0.000 0.000  
2 0.000 0.057 0.902 0.041 0.000  
3 0.000 0.000 0.038 0.914 0.048  
4 0.000 0.003 0.000 0.042 0.955  
ergodic matrix:  
 0 1 2 3 4  
0 4.934 16.095 48.695 118.064 194.493  
1 63.323 5.311 32.600 101.969 178.398  
2 114.113 50.789 5.030 69.369 145.798  
3 160.206 96.883 52.508 5.046 76.429  
4 176.147 112.824 73.581 29.198 4.715  
Steady\_state: [0.203 0.188 0.199 0.198 0.212]



#### 4）莫兰指数 （Moran’s Is）

经典离散型马尔可夫链并未考虑空间邻里关系，即一个空间单元的状态转移受到该单元邻里单元的影响情况并未考虑。下述定义了gdf\_plot\_annotate\_subplots()函数，可以打印多个时间节点观测数值，从图表结果来看不难发现，按分位数划分的区间分类，在各年均具有明显的集聚特征，表明空间邻里单元可能会影响到目标单元的状态转移概率。

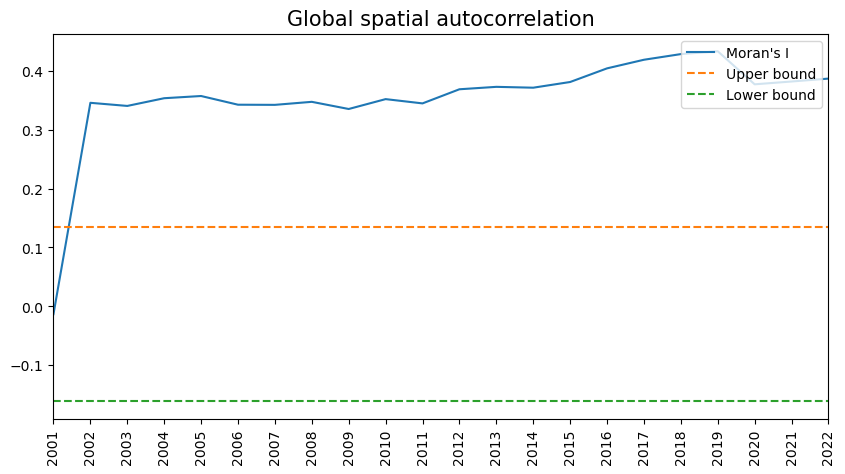
def gdf\_plot\_annotate\_subplots(gdf\_,value\_column,annotate\_column,ts\_column,ts\_lst,ncols=3,\*\*setting):   
 '''  
 打印多个时间点（分组）地图  
  
 Parameters  
 ----------  
 gdf\_ : GeoDataFrame  
 地理时空数据.  
 value\_column : string  
 观测值列名.  
 annotate\_column : string   
 标注列名.  
 ts\_column : string  
 时间(分组)列名.  
 ts\_lst : list(string)  
 时间点列表.  
 ncols : int, optional  
 图表子图列数. The default is 3.  
 \*\*setting : TYPE  
 图表样式参数，默认为：  
 setting\_dict=dict(annotate\_fontsize=8,  
 figsize=(10,10),   
 legend\_position="right",  
 legend\_size="5%",  
 legend\_pad=0.1,  
 legend\_bbox\_to\_anchor=(1, 1),  
 cmap='OrRd',  
 labelsize=8,  
 scheme=None,  
 k=5,  
 categorical=False,  
 ).  
  
 Returns  
 -------  
 None.  
  
 '''   
 import matplotlib.pyplot as plt  
 from mpl\_toolkits.axes\_grid1 import make\_axes\_locatable  
   
 gdf=gdf\_.copy(deep=True)  
 setting\_dict=dict(annotate\_fontsize=8,  
 figsize=(10,10),   
 legend\_position="right",  
 legend\_size="5%",  
 legend\_pad=0.1,  
 legend\_bbox\_to\_anchor=(1, 1),  
 cmap='OrRd',  
 labelsize=8,  
 scheme=None,  
 k=5,  
 categorical=False,  
 )  
 setting\_dict.update(setting)  
 gdf["index"]=gdf.index  
   
 nrows=len(ts\_lst)//ncols  
 fig, axes=plt.subplots(nrows=nrows, ncols=ncols,figsize=setting\_dict["figsize"])  
   
 ts\_gdfs=[]  
 for ts in ts\_lst:  
 ts\_gdfs.append(gdf[gdf[ts\_column]==ts])  
   
 for i in range(nrows):  
 for j in range(ncols):  
 ax=axes[i,j]   
 idx=i\*ncols+j  
 if idx<len(ts\_lst):  
 ts\_gdf=ts\_gdfs[idx]   
 divider=make\_axes\_locatable(ax)   
 if setting\_dict["scheme"]:  
 ts\_gdf.plot(column=value\_column,scheme=setting\_dict["scheme"], k=setting\_dict["k"],ax=ax,legend=True,cmap=setting\_dict["cmap"],legend\_kwds={'bbox\_to\_anchor':setting\_dict["legend\_bbox\_to\_anchor"]})   
 elif setting\_dict["categorical"]:  
 ts\_gdf.plot(column=value\_column,categorical=True,ax=ax,legend=True,cmap=setting\_dict["cmap"],edgecolor='white',legend\_kwds={'bbox\_to\_anchor':setting\_dict["legend\_bbox\_to\_anchor"]})   
 else:   
 cax=divider.append\_axes(setting\_dict["legend\_position"], size=setting\_dict["legend\_size"], pad=setting\_dict["legend\_pad"]) # 配置图例参数  
 ts\_gdf.plot(column=value\_column,scheme=setting\_dict["scheme"], k= setting\_dict["k"],ax=ax,cax=cax,legend=True,cmap=setting\_dict["cmap"])   
 ts\_gdf.apply(lambda x: ax.annotate(text=x[annotate\_column], xy=x.geometry.centroid.coords[0], ha='center',fontsize=setting\_dict["annotate\_fontsize"]),axis=1) # 增加标注  
 ax.tick\_params(axis='both', labelsize=setting\_dict["labelsize"])  
 ax.set\_title(ts\_lst[idx])  
   
 plt.tight\_layout()  
 plt.show()  
   
crimes\_Y\_gdf['ts\_sft']=crimes\_Y\_gdf['ts'].apply(lambda x:x.strftime('%Y'))  
gdf\_plot\_annotate\_subplots(crimes\_Y\_gdf,"Y\_sum","area\_numbe",ts\_column="ts\_sft",ts\_lst=['2001','2005','2009','2014','2018','2022'],annotate\_fontsize=10,scheme="Quantiles",k=5,figsize=(30,15),legend\_bbox\_to\_anchor=(0.4, 0.2))



为进一步量化证明空间单元邻里测量值的相关性，使用全局空间自相关衡量，定义GIs4panel\_data\_plot()函数打印各年Moran’s I指数，可以发现，2002年至2022年，Moran’s I值为正高于0.3，表明各个社区的犯罪报告数并不是相互独立的，单元邻里背景区域可能对该单元测量值的动态形成有重要影响。

def GIs4panel\_data\_plot(gdf,value\_column,group\_column,ts\_column,\*\*setting):  
 '''  
 打印各个时间点上各分组测量值全局空间自相关指数  
  
 Parameters  
 ----------  
 gdf : GeoDataFrame  
 时空地理数据.  
 value\_column : string  
 测量值（观测值）列名.  
 group\_column : string  
 分组列名.  
 ts\_column : string  
 时间列名.  
 \*\*setting : TYPE  
 图表打印样式，默认值为：  
 setting\_dict=dict(figsize=(10,5),   
 fontsize=15,  
 rotation=90,  
 ).  
  
 Returns  
 -------  
 wr : libpysal.weights.contiguity.Rook  
 空间权重.  
  
 '''   
 import libpysal.weights as LW  
 from esda.moran import Moran  
 import numpy as np  
 import matplotlib.pyplot as plt  
   
 setting\_dict=dict(figsize=(10,5),   
 fontsize=15,  
 rotation=90,  
 )   
 setting\_dict.update(setting)  
   
 grouped\_gdf=gdf.groupby(ts\_column)  
 mits=[]  
 names=[]  
 for name,group in grouped\_gdf:  
 wr=LW.contiguity.Rook.from\_dataframe(group)  
 wr.transform='R'  
 mits.append(Moran(group[value\_column],wr))  
 names.append(name)  
   
 print(names)  
 res=np.array([(mi.I, mi.EI, mi.seI\_norm, mi.sim[0]) for mi in mits])  
 fig, ax=plt.subplots(nrows=1,ncols=1,figsize=setting\_dict['figsize'])  
 ax.plot(names, res[:,0], label='Moran\'s I')  
  
 ax.plot(names, res[:,1]+1.96\*res[:,2], label='Upper bound',linestyle='dashed')  
 ax.plot(names, res[:,1]-1.96\*res[:,2], label='Lower bound',linestyle='dashed')  
 ax.set\_title("Global spatial autocorrelation",fontdict={'fontsize':setting\_dict['fontsize']})  
 ax.set\_xlim([names[0],names[-1]])  
 ax.legend()  
 plt.xticks(rotation=setting\_dict['rotation'])  
   
 return wr  
  
wr=GIs4panel\_data\_plot(crimes\_Y\_gdf,"Y\_sum","area\_numbe",'ts\_sft')

['2001', '2002', '2003', '2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014', '2015', '2016', '2017', '2018', '2019', '2020', '2021', '2022']



#### 5）空间马尔科夫 （Spatial Markov）

空间马尔科夫是经典马尔科夫的延申，是对时空地理数据状态转化在空间维度上更全面的分析[3]。空间马尔科夫的转移概率是否受到邻里单元的影响已有研究且量化。空间马尔科夫不是估计一个转移概率矩阵，而是需要估计个转移概率矩阵，每个矩阵都是以前一时期邻里单元背景为条件。邻里单元背景条件通常由空间滞后邻里单元测量值加权平均值确定，公式为： wr,srstrys,tst时刻的观测值。

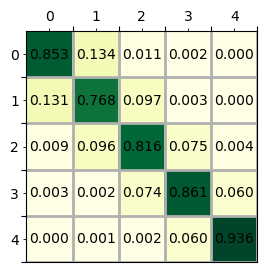
PySAL-giddy库提供了Spatial\_Markov空间马尔科夫计算方法，该计算方法整合和了mapclassify库提供的分类方法，因此可以直接输入原始连续观测值，而不用事先计算分类，例如前文用到的5分位数区间划分等。打印转移概率矩阵，其结果与经典马尔科夫计算结果存在差异，为考虑邻里单元空间权重影响后的结果。

crimes\_M\_gdf

|  | **ts** | **M\_sum** | **area\_numbe** | **geometry** | **ts\_sft** |
| --- | --- | --- | --- | --- | --- |
| **0** | 2001-01-31 | 11.0 | 1 | POLYGON ((-87.65456 41.99817, -87.65574 41.998... | 200101 |
| **1** | 2001-02-28 | 3.0 | 1 | POLYGON ((-87.65456 41.99817, -87.65574 41.998... | 200102 |
| **2** | 2001-03-31 | 4.0 | 1 | POLYGON ((-87.65456 41.99817, -87.65574 41.998... | 200103 |
| **3** | 2001-04-30 | 5.0 | 1 | POLYGON ((-87.65456 41.99817, -87.65574 41.998... | 200104 |
| **4** | 2001-05-31 | 3.0 | 1 | POLYGON ((-87.65456 41.99817, -87.65574 41.998... | 200105 |
| **...** | ... | ... | ... | ... | ... |
| **20231** | 2022-07-31 | 216.0 | 77 | POLYGON ((-87.65456 41.99817, -87.65456 41.998... | 202207 |
| **20232** | 2022-08-31 | 268.0 | 77 | POLYGON ((-87.65456 41.99817, -87.65456 41.998... | 202208 |
| **20233** | 2022-09-30 | 261.0 | 77 | POLYGON ((-87.65456 41.99817, -87.65456 41.998... | 202209 |
| **20234** | 2022-10-31 | 283.0 | 77 | POLYGON ((-87.65456 41.99817, -87.65456 41.998... | 202210 |
| **20235** | 2022-11-30 | 173.0 | 77 | POLYGON ((-87.65456 41.99817, -87.65456 41.998... | 202211 |

20236 rows × 5 columns

import giddy  
import libpysal.weights as LW  
import numpy as np  
  
val\_array=val\_array\_M  
crimes\_M\_gdf['ts\_sft']=crimes\_M\_gdf['ts'].apply(lambda x:x.strftime('%Y'))  
  
#wr=LW.contiguity.Rook.from\_dataframe(crimes\_M\_gdf.groupby('ts\_sft').get\_group('2001'))  
wr=LW.contiguity.Rook.from\_dataframe(community\_area\_gdf)  
wr.transform='R'  
r\_val\_array=(val\_array.T / val\_array.mean(axis=1))  
sm=giddy.markov.Spatial\_Markov(r\_val\_array,wr,fixed=True,k=5,m=5, fill\_empty\_classes=True ) # spatial\_markov instance   
colorplot\_2d\_array\_with\_labels(sm.p.round(3),sm.classes,figsize=(3,3))

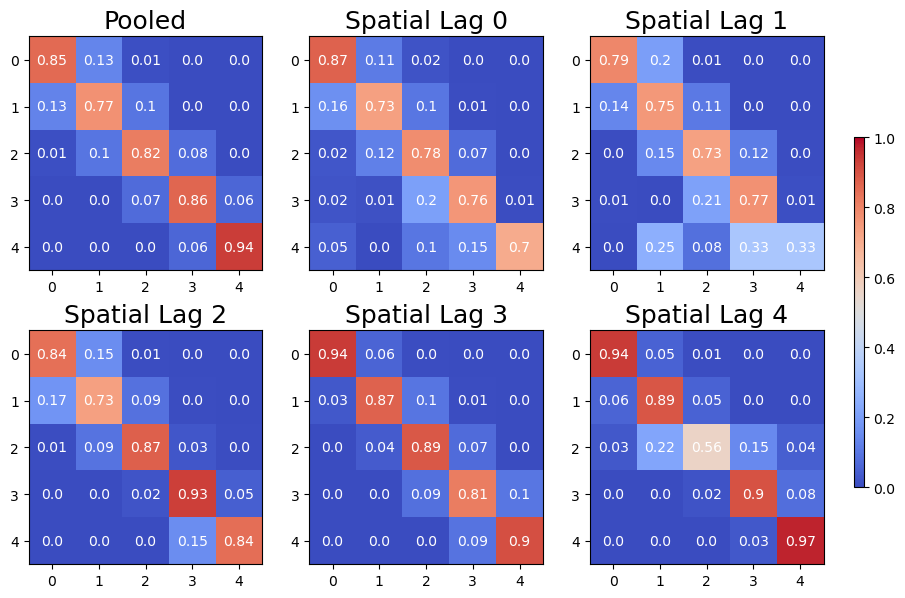


PySAL-giddy提供了P或者summary方法可以提取分析查看目标单元的邻里单元处于不同状态（不同分位数，由低值到高值标签0~4）下，目标单元的转移概率，例如图表中的“Spatial Lag 0”，为邻里单元状态为分位数标签0时不同状态转移概率矩阵，可以发现位于的状态转移概率为0.87，说明当邻里单元观测值为状态标签0，即较低值时，目标单元如果状态标签也为0，处于较低值时，目标单元保持自身较低值的状态转移概率很高。但也注意到，的状态转移概率为0.7，一定程度上显示当邻里单元观测值为状态标签0时，目标单元如果状态标签为4较高值时，目标单元保持自身状态的概率较之目标单元为0的状态时低，即有部分会向状态3、2等低值状态转化；同样，从图表中的“Spatial Lag 4”可以观察到如果邻里单元观测值为标签4，较高值时，目标单元也为标签4的较高值，则目标单元保持自身较高值的状态转移概率很高为0.97，但如果邻里单元观测值为标签3时，对应图“Spatial Lag 3”，则目标单元为标签4保持自身的概率降至0.9。同时，注意到，在“Spatial Lag 4”中，的转移概率为0.94，接近，的转移概率0.97，但仅为0.56，这一定程度上表明，当邻里单元状态为高值时，目标单元处于状态高低区域保持自身的状态的概率都很高，但处于中间区域，例如状态2，向1和3状态都有一定转移。

sm.summary() # sm.P

--------------------------------------------------------------  
 Spatial Markov Test   
--------------------------------------------------------------  
Number of classes: 5  
Number of transitions: 20174  
Number of regimes: 5  
Regime names: LAG0, LAG1, LAG2, LAG3, LAG4  
--------------------------------------------------------------  
 Test LR Chi-2  
 Stat. 1073.791 2361.386  
 DOF 76 76  
p-value 0.000 0.000  
--------------------------------------------------------------  
P(H0) C0 C1 C2 C3 C4  
 C0 0.853 0.134 0.011 0.002 0.000  
 C1 0.131 0.768 0.097 0.003 0.000  
 C2 0.009 0.096 0.816 0.075 0.004  
 C3 0.003 0.002 0.074 0.861 0.060  
 C4 0.000 0.001 0.002 0.060 0.936  
--------------------------------------------------------------  
P(LAG0) C0 C1 C2 C3 C4  
 C0 0.870 0.110 0.017 0.002 0.000  
 C1 0.164 0.731 0.100 0.005 0.000  
 C2 0.024 0.122 0.779 0.072 0.002  
 C3 0.020 0.014 0.203 0.756 0.007  
 C4 0.050 0.000 0.100 0.150 0.700  
--------------------------------------------------------------  
P(LAG1) C0 C1 C2 C3 C4  
 C0 0.791 0.199 0.007 0.003 0.000  
 C1 0.138 0.752 0.109 0.001 0.001  
 C2 0.004 0.146 0.730 0.116 0.004  
 C3 0.008 0.004 0.210 0.767 0.011  
 C4 0.000 0.250 0.083 0.333 0.333  
--------------------------------------------------------------  
P(LAG2) C0 C1 C2 C3 C4  
 C0 0.840 0.152 0.006 0.001 0.000  
 C1 0.173 0.734 0.087 0.004 0.001  
 C2 0.007 0.091 0.871 0.026 0.005  
 C3 0.000 0.002 0.020 0.931 0.047  
 C4 0.000 0.000 0.002 0.154 0.844  
--------------------------------------------------------------  
P(LAG3) C0 C1 C2 C3 C4  
 C0 0.938 0.058 0.004 0.000 0.000  
 C1 0.029 0.867 0.096 0.008 0.000  
 C2 0.002 0.035 0.889 0.073 0.002  
 C3 0.001 0.000 0.085 0.814 0.100  
 C4 0.000 0.000 0.004 0.094 0.902  
--------------------------------------------------------------  
P(LAG4) C0 C1 C2 C3 C4  
 C0 0.941 0.052 0.007 0.000 0.000  
 C1 0.056 0.893 0.052 0.000 0.000  
 C2 0.029 0.221 0.559 0.147 0.044  
 C3 0.000 0.001 0.016 0.901 0.082  
 C4 0.000 0.000 0.001 0.029 0.970  
--------------------------------------------------------------

def spatial\_markov\_lag(sm,\*\*args):  
 '''  
 打印空间马尔科夫状态滞后图表  
  
 Parameters  
 ----------  
 sm : giddy.markov.Spatial\_Markov  
 空间马尔科夫链.  
 \*\*args : TYPE  
 图表打印样式，默认值为：  
 setting=dict(figsize=(10,7),  
 cmap="coolwarm",  
 subplots\_adjust\_right=0.92,  
 subplots\_rows\_cols=(2,3)  
 ).  
  
 Returns  
 -------  
 None.  
  
 '''   
 import matplotlib.pyplot as plt  
   
 setting=dict(figsize=(10,7),  
 cmap="coolwarm",  
 subplots\_adjust\_right=0.92,  
 subplots\_rows\_cols=(2,3)  
 )  
   
 fig, axes=plt.subplots(setting['subplots\_rows\_cols'][0],setting['subplots\_rows\_cols'][1],figsize=setting['figsize'])   
 for i in range(2):  
 for j in range(3):  
 ax=axes[i,j]  
 if i==0 and j==0:  
 p\_temp=sm.p  
 im=ax.imshow(p\_temp,cmap=setting['cmap'],vmin=0, vmax=1)  
 ax.set\_title("Pooled",fontsize=18)   
 else:  
 p\_temp=sm.P[i\*3+j-1]  
 im=ax.imshow(p\_temp,cmap=setting['cmap'],vmin=0, vmax=1)  
 ax.set\_title("Spatial Lag %d"%(i\*3+j-1),fontsize=18)   
 for x in range(len(p\_temp)):  
 for y in range(len(p\_temp)):  
 text=ax.text(y, x, round(p\_temp[x, y], 2),  
 ha="center", va="center", color="w")  
  
 fig.subplots\_adjust(right=setting['subplots\_adjust\_right'])  
 cbar\_ax=fig.add\_axes([0.95, 0.228, 0.01, 0.5])  
 fig.colorbar(im, cax=cbar\_ax)  
 plt.show()  
   
spatial\_markov\_lag(sm)



sm.S方法可以估计不同状态空间滞后的平稳状态分布，从结果可以得知，各行为邻里单元所处状态（由低值到高值标签0-4的不同分位数），各列为目标单元所处状态（由低值到高值标签0-4的不同分位数），例如对于第1行第1个值，邻里单元状态为0（低值），目标单元处于状态0长期分布有0.407的比例，第1行第2个值，目标单元处于状态1长期分布有0.274的比例，依次类推，对于第1行最后一个值，目标单元处于状态4时长期分布有0.004的比例，每一行的比例之和为1；同样，对于第5行最后一个值，为邻里单元状态为4（高值）时，目标单元状态为4时，长期分布占0.588的比例等。可以发现，长期平稳状态下，邻里单元观测值为低值时，目标单元中观测值也处于低值的比例较高；同样，邻里单元观测值为高值时，目标单元中观测值也处于高值的比例较高。

sm.S.round(3)

array([[0.407, 0.276, 0.233, 0.081, 0.004],  
 [0.243, 0.354, 0.259, 0.139, 0.004],  
 [0.21 , 0.186, 0.187, 0.315, 0.102],  
 [0.062, 0.107, 0.302, 0.259, 0.27 ],  
 [0.096, 0.092, 0.021, 0.203, 0.588]])

sm.F用于计算邻里单元处于不同状态下状态转移平均所需时间，例如当邻里单元为状态0时（第1组），目标单元状态为0时，回到状态0即需要2.485年，而状态0到状态1即需要9.334年，如果要到达状态4即则需要961.611年；同样，当邻里单元为状态4时（最后一组），目标单元为状态0，对于的状态转化只需要113.74年，较之处于邻里单元状态为0的条件，的转换961.611年而言有较大的缩短；而，由也仅需要1.701年。

print(sm.F.round(3))

[[[ 2.458 9.334 23.015 63.717 961.611]  
 [ 10.251 3.624 17.879 58.997 956.69 ]  
 [ 16.576 10.219 4.3 44.468 940.537]  
 [ 18.99 13.813 7.309 12.335 921.161]  
 [ 18.354 15.202 10.824 28.775 281.308]]  
  
 [[ 4.109 5.301 18.581 43.479 366.569]  
 [ 14.389 2.822 14.403 39.711 362.524]  
 [ 23.297 9.887 3.864 26.116 350.607]  
 [ 26.688 13.867 5.613 7.182 338.518]  
 [ 23.152 9.67 9.708 19.656 233.687]]  
  
 [[ 4.763 8.045 30.582 101.921 146.671]  
 [ 21.808 5.371 25.058 97.031 141.689]  
 [ 48.106 28.492 5.347 82.26 127.081]  
 [104.002 84.241 59.329 3.176 55.402]  
 [109.592 89.833 64.869 7.621 9.805]]  
  
 [[ 16.184 21.342 30.532 47.755 73.954]  
 [241.123 9.343 15.496 32.496 58.699]  
 [297.048 72.787 3.308 20.118 46.264]  
 [316.262 94.698 22.812 3.863 26.934]  
 [325.743 104.071 32.149 11.055 3.703]]  
  
 [[ 10.4 34.859 46.06 104.692 113.74 ]  
 [142.448 10.885 33.134 91.767 100.815]  
 [275.786 153.298 48.063 58.633 67.681]  
 [460.227 336.555 194.31 4.921 22.271]  
 [488.86 365.22 222.683 35.03 1.701]]]

对于条件型空间依赖性的检验包括似然比（Likelihood Ratio，LR）检验和X²检验[4]，以及基于信息论的检验[5]，对于前两个检验，可以用giddy库提供的如下方法计算，获得统计数据、DOF和P值。

从计算结果来看，LR检验结果为111.320，p-value值为0；χ2检验结果为419.927，p-value值为0，因此拒绝条件空间独立性的无效假设。

giddy.markov.Homogeneity\_Results(sm.T).summary()

--------------------------------------------------  
 Markov Homogeneity Test   
--------------------------------------------------  
Number of classes: 5  
Number of transitions: 1617  
Number of regimes: 5  
Regime names: 0, 1, 2, 3, 4  
--------------------------------------------------  
 Test LR Chi-2  
 Stat. 111.320 419.927  
 DOF 42 42  
p-value 0.000 0.000  
--------------------------------------------------  
P(H0) 0 1 2 3 4  
 0 0.887 0.110 0.003 0.000 0.000  
 1 0.074 0.839 0.084 0.003 0.000  
 2 0.000 0.053 0.875 0.072 0.000  
 3 0.000 0.000 0.044 0.881 0.075  
 4 0.000 0.003 0.000 0.046 0.951  
--------------------------------------------------  
P(0) 0 1 2 3 4  
 0 0.880 0.113 0.007 0.000 0.000  
 1 0.099 0.736 0.154 0.011 0.000  
 2 0.000 0.100 0.786 0.114 0.000  
 3 0.000 0.000 0.136 0.818 0.045  
 4 0.000 1.000 0.000 0.000 0.000  
--------------------------------------------------  
P(1) 0 1 2 3 4  
 0 0.831 0.169 0.000 0.000 0.000  
 1 0.066 0.844 0.090 0.000 0.000  
 2 0.000 0.095 0.784 0.122 0.000  
 3 0.000 0.000 0.186 0.744 0.070  
 4 0.000 0.000 0.000 0.000 0.000  
--------------------------------------------------  
P(2) 0 1 2 3 4  
 0 0.937 0.063 0.000 0.000 0.000  
 1 0.143 0.833 0.024 0.000 0.000  
 2 0.000 0.030 0.955 0.015 0.000  
 3 0.000 0.000 0.000 0.939 0.061  
 4 0.000 0.000 0.000 0.081 0.919  
--------------------------------------------------  
P(3) 0 1 2 3 4  
 0 0.941 0.059 0.000 0.000 0.000  
 1 0.021 0.958 0.021 0.000 0.000  
 2 0.000 0.009 0.943 0.047 0.000  
 3 0.000 0.000 0.042 0.845 0.113  
 4 0.000 0.000 0.000 0.090 0.910  
--------------------------------------------------  
P(4) 0 1 2 3 4  
 0 0.955 0.045 0.000 0.000 0.000  
 1 0.000 1.000 0.000 0.000 0.000  
 2 0.000 0.000 1.000 0.000 0.000  
 3 0.000 0.000 0.000 0.928 0.072  
 4 0.000 0.000 0.000 0.024 0.976  
--------------------------------------------------

基于信息论的检验调用kullback方法计算，结果与LR和χ2检验结果同，观察到检验统计量为162.419，其p-value为1.484809997176484e-07，拒绝条件空间独立性的无效假设。

print(giddy.markov.kullback(sm.T))

{'Conditional homogeneity': 162.41979588493268, 'Conditional homogeneity dof': 80, 'Conditional homogeneity pvalue': 1.484809997176484e-07}

#### 6）局部空间自相关马尔科夫（LISA Markov）

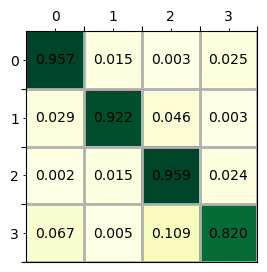
giddy提供了LISA\_Markov方法，将局部空间自相关，高-高（HH，0）、低-高（LH，1）、低-低（LL，2）和高-低（HL，3）作为4种状态，计算这4种状态之间的转移矩阵、转移概率矩阵和平稳状态转移概率等。从下述计算结果，观察转移概率矩阵的对角线估计值，可以发现HL（3）概率值为0.820，相对HH（0）、LH（1）和LL（2）值低，即具有相对较高的流动性，其它3种状态则具有相对较高的稳定性。

为了检验目标单元与其邻里单元间的动态关系，使用lm.chi\_2)方法进行χ2的独立性检验，其p-value等于0，拒绝原条件空间独立性的无效假设，即目标单元和邻里单元对犯罪报告记录局部空间自相关4种状态的转变存在空间的影响。

lm=giddy.markov.LISA\_Markov(val\_array.T, wr)  
markov\_info\_print(lm,decimal=3)

--------------------------------------------------  
Unique classes: [1 2 3 4]  
Transition matrix:  
 1 2 3 4  
1 5228.0 80.0 15.0 138.0  
2 87.0 2764.0 139.0 9.0  
3 21.0 143.0 9232.0 227.0  
4 140.0 10.0 227.0 1714.0  
Transition probability matrix:  
 1 2 3 4  
1 0.957 0.015 0.003 0.025  
2 0.029 0.922 0.046 0.003  
3 0.002 0.015 0.959 0.024  
4 0.067 0.005 0.109 0.820  
ergodic matrix:  
 1 2 3 4  
1 3.497 73.854 48.897 46.777  
2 61.467 6.774 31.804 58.854  
3 78.806 73.367 2.158 48.648  
4 54.628 77.148 24.548 9.695  
Steady\_state: [0.286 0.148 0.463 0.103]

colorplot\_2d\_array\_with\_labels(lm.p.round(3),sm.classes,figsize=(3,3))

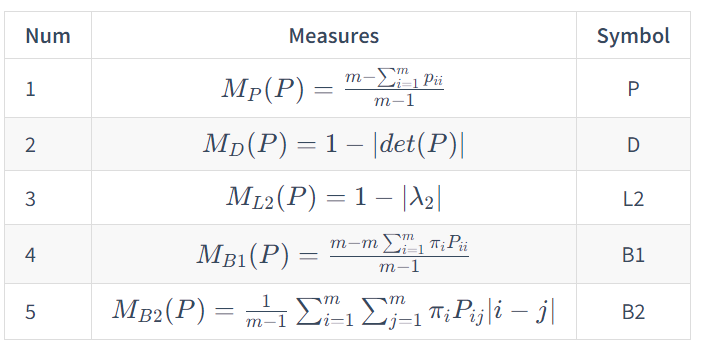


print(lm.chi\_2)

(18720.987813562097, 0.0, 9)

### 2.7.4.3 流动性测量（Mobility\_measures）[6]

流动性可以看作是一种重新排序的现象，即各单元的状态转移；同时，可以认为只要各单元远离之前的状态就会发生流动性，前者称为绝对流动性，后者称为相对流动性。giddy库提供了5种基于马尔可夫的流动性估计方法，如下：



式中，*π*为初始状态分布。对于任何具有准最大对角线的转移概率矩阵，所有上述流动性度量的值域为[0,1]，0表示不流动，1表示流动。例如转移概率矩阵的对象线均为1，则流动性为0，表示各单元状态保持自身状态不变；如果转移概率矩阵每一行值相同，即单元每一状态转移为任何其它状态的概率相同，流动性1，表示完全的流动性。这5种衡量标准试图从不同方面揭示流动性模式，衡量的均值和标准差不同，因此彼此之间不具可比性。

基于以芝加哥犯罪报告记录5分位数为单元状态计算马尔可夫链，即上文计算所得的m5变量值，用mobility方法计算流动性。从计算结果来看值普遍较小，一定程度上表明犯罪报告记录的流动性小，即各个单元地区基本会保持自身的状态，而不容易向其它状态转移。

from giddy import markov,mobility  
  
m\_P=mobility.markov\_mobility(m5.p, measure="P")  
m\_D=mobility.markov\_mobility(m5.p, measure="D")  
m\_L2=mobility.markov\_mobility(m5.p, measure = "L2")  
pi=np.array([0.1,0.2,0.2,0.4,0.1])  
m\_B1=mobility.markov\_mobility(m5.p, measure = "B1", ini=pi)  
m\_B2=mobility.markov\_mobility(m5.p, measure = "B2", ini=pi)  
  
print(f"mobility measure Shorrock1: {m\_P};\nShorroks2: {m\_D};\nSommers and Conlisk: {m\_L2};\nBartholomew1: {m\_B1};\nBartholomew2: {m\_B2}")

mobility measure Shorrock1: 0.10543656065612206;  
Shorroks2: 0.36697357812338227;  
Sommers and Conlisk: 0.019698993481535365;  
Bartholomew1: 0.11358674661054002;  
Bartholomew2: 0.022866158845917545

注释（Notes）：

① MODIS Grids，（<https://modis-land.gsfc.nasa.gov/MODLAND_grid.html>）。

② Table of Tile Bounding Coordinates (10 deg tiles)，（<https://modis-land.gsfc.nasa.gov/pdf/sn_bound_10deg.txt>）。

③ Table of Tile G-ring Coordinates (10 deg tiles)，（<https://modis-land.gsfc.nasa.gov/pdf/sn_gring_10deg.txt>）。

④ General Cartographic Transformation Package (GCTP) projection parameters and other tile infomation，（<https://modis-land.gsfc.nasa.gov/GCTP.html>）。

⑤ Table of Tile Bounding Coordinates，（<https://modis-land.gsfc.nasa.gov/pdf/MODISPolarGridTiles.pdf>）。

⑥ lat/long bounding (G-ring) coordinates，（<https://modis-land.gsfc.nasa.gov/lat_long_tiles.txt>）。

⑦ MCD12Q1\_v006数据集下载地址，（<https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.006/>）。

⑧ Web crawling with Python，（<https://www.scrapingbee.com/blog/crawling-python/>）。

⑨ plot\_colortable()绘制一个matplotlib支持命名颜色的列表函数，（<https://matplotlib.org/3.1.1/gallery/color/named_colors.html>）。

⑩ rioxarray，rasterio xarray extension（<https://github.com/corteva/rioxarray>）。

⑪ 2015年中国省级行政边界数据，来自于中国科学院资源环境科学与数据中心（<https://www.resdc.cn/data.aspx?DATAID=200>）。

⑫ Xarray库，（<https://docs.xarray.dev/en/stable/index.html>）。

⑬ netCDF（Network Common Data Form），（<https://en.wikipedia.org/wiki/NetCDF>）。

⑭ Crimes - 2001 to present，该数据集反映了2001年至今发生在芝加哥市的报告犯罪事件（谋杀除外），减去最近的七天。（<https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present-Dashboard/5cd6-ry5g>）。

⑮ Time series / date functionality：Offset aliases，一些字符串别名被赋予有用的普通时间序列频率,称为偏移别名（<https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html#offset-aliases>）。

⑯ PySAL-giddy，Giddy是一个开源的Python库，用于分析纵向空间数据的动态变化（<https://pysal.org/notebooks/explore/giddy/intro.html>）。

参考文献（References）:

[1] Damien Sulla-Menashe and Mark A Friedl. User Guide to Collection 6 MODIS Land Cover (MCD12Q1 and MCD12C1) Product.May 14, 2018. <https://lpdaac.usgs.gov/products/mcd12q1v006/>

[2] Spatially Explicit Markov Methods，<http://pysal.org/notebooks/explore/giddy/Markov_Based_Methods.html>.

[3] Rey, S. J. 2001. “Spatial Empirics for Economic Growth and Convergence.” Geographical Analysis 33 (3). Wiley Online Library: 195–214.

[4] Bickenbach, F., and E. Bode. 2003. “Evaluating the Markov Property in Studies of Economic Convergence.” International Regional Science Review 26 (3): 363–92.

[5] Kullback, S., M. Kupperman, and H. H. Ku. 1962. “Tests for Contingency Tables and Markov Chains.” Technometrics: A Journal of Statistics for the Physical, Chemical, and Engineering Sciences 4 (4). JSTOR: 573–608.

[6] Measures of Income Mobility ，<http://pysal.org/notebooks/explore/giddy/Mobility_measures.html>.