第 6 节：GEE 的数据类型(Geometry，Feature，Feature Collection)

本节介绍 GEE 中的矢量信息。总体上 Geometry 可以看作是“形状”，Feature 可以看作是“带属性的形状”，而 Feature Collection 则是“一个或多个 Feature 的集合”。有关矢量的操作整体上可以分为两类：一类是涉及形状的操作，另一类是关于属性的操作。通过本节的学习，我们将掌握矢量数据的创建、结构调整、形状调整和属性调整。

6.1 Geometry

Geometry 由点、线和面组成。因为线和面本质都是由点构成的，因此可以认为有关 Geometry 的操作都是在确定其组成点的坐标。

下边介绍单点和多点的创建，代码及操作效果如下所示：

```
var Forbidden_City = ee.Geometry.Point(116.3968,39.9186)

Map.centerObject(Forbidden_City,15)
print(Forbidden_City)
Map.addLayer(Forbidden_City)
```

```
var Forbidden_City = ee.Geometry.MultiPoint(
            [[116.3921, 39.9224],
             [116.4014, 39.9227],
             [116.4017, 39.9138],
             [116.3929, 39.9135]                    ]);

Map.centerObject(Forbidden_City,15)
print(Forbidden_City)
Map.addLayer(Forbidden_City)
```

这里需要注意两点：第一，利用代码创建 Geometry(点、线和面)的效率很低，在实际操作中通常利用 (手绘工具)进行操作。对于本例来说，对应的操作是：首先点击 ，等鼠标变成十字光标后，再点击例子中的点所在位置，最后在代码区上部的 Imports 部分会出现 var geometry: Point (116.40, 39.92) ，点击geometry并重命名为“Forbidden_City”即可。利用手绘工具进行的操作与代码方式引入的点是完全等效的。第二，代码中的 Map.CenterObject(,)命令的功能是将底图中心设置为选定位置，该命令的第一个参数是“位置对象”，第二个参数是缩放级别。另外，Map.addLayer()的功能是将图像数据加载到底图上。最后，在没有特别指明的情况下，利用代码创建 Geometry 的方式均可以由手绘工具通过点击实现，并且在实际操作中推荐利用手绘工具创建 Geometry 以提高效率。

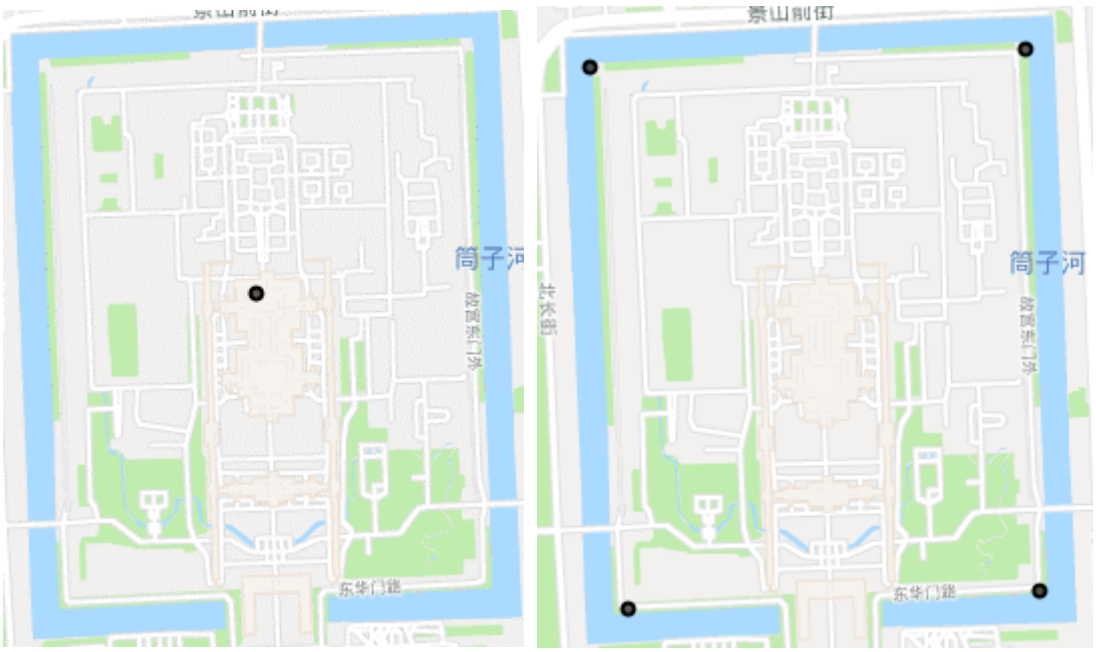


图 6.1 点的创建

下边介绍线段和多线的创建，代码及操作效果如下所示：

```
var Forbidden_City = ee.Geometry.LineString(
            [[116.39234040929296, 39.91321497621612],
             [116.39195417119481, 39.92276018488313],
             [116.40186761571385, 39.9230234821268],
             [116.40216802312352, 39.91354414349019]]);

Map.centerObject(Forbidden_City,15)
print(Forbidden_City)
Map.addLayer(Forbidden_City)
```

```
var Forbidden_City = ee.Geometry.MultiLineString([
                [[116.39367078496434, 39.921344944851143],
                 [116.39899228764989, 39.92173989850053]],

                [[116.39401410771825, 39.91953471147776],
                 [116.39693235112645, 39.92085124958646]],

                [[116.39628862096288, 39.91920557299557],
                 [116.39873479558446, 39.91752694212849]]       ]);

Map.centerObject(Forbidden_City)
print(Forbidden_City)
Map.addLayer(Forbidden_City)
```

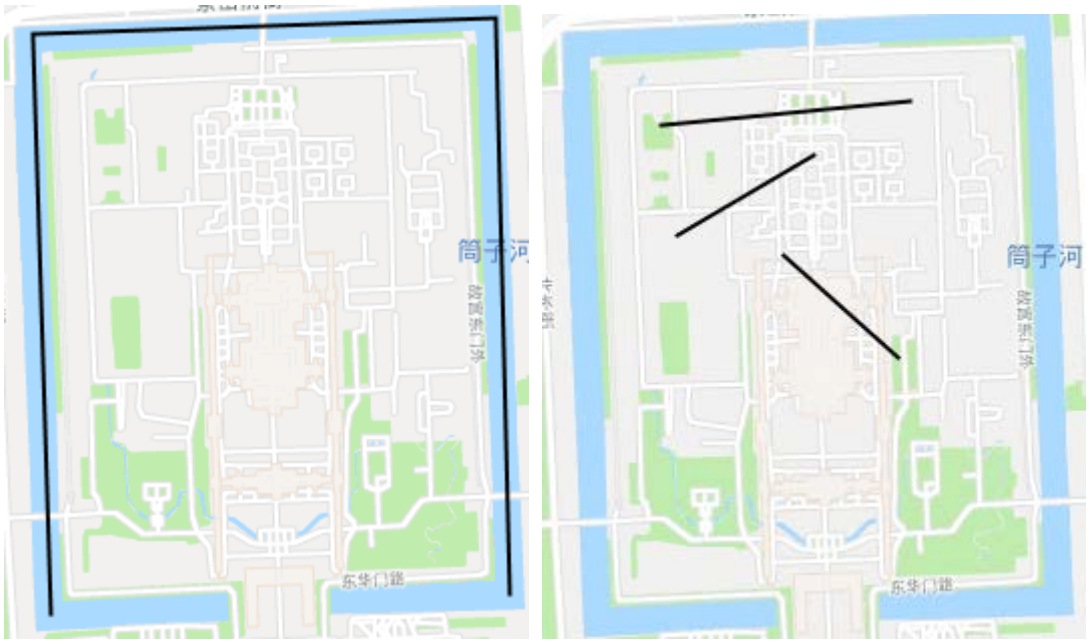


图 6.2 线段和多线的创建

下边介绍单线环和多线环的创建，代码及操作效果如下所示：

```
var Forbidden_City = ee.Geometry.LinearRing(
            [[116.39242623998143, 39.91321497621612],
             [116.39195417119481, 39.92269436041402],
             [116.40178178502538, 39.9230234821268],
             [116.40212510777928, 39.91357706013058],
             [116.39242623998143, 39.91321497621612]]);

Map.centerObject(Forbidden_City)
print(Forbidden_City)
Map.addLayer(Forbidden_City)
```

```
var Forbidden_City =ee.Geometry.MultiLineString(
            [[[116.39320681392098, 39.921786784372976],
              [116.39337847529794, 39.92040443758097],
              [116.39578173457528, 39.92070065709998],
              [116.39556715785409, 39.92188552236169],
              [116.39320681392098, 39.921786784372976]],
             [[116.39801333247567, 39.91918663276216],
              [116.39548132716561, 39.91777131887048],
              [116.4006311684742, 39.91747508667966],
              [116.39801333247567, 39.91918663276216]]]);

Map.centerObject(Forbidden_City)
print(Forbidden_City)
Map.addLayer(Forbidden_City)
```

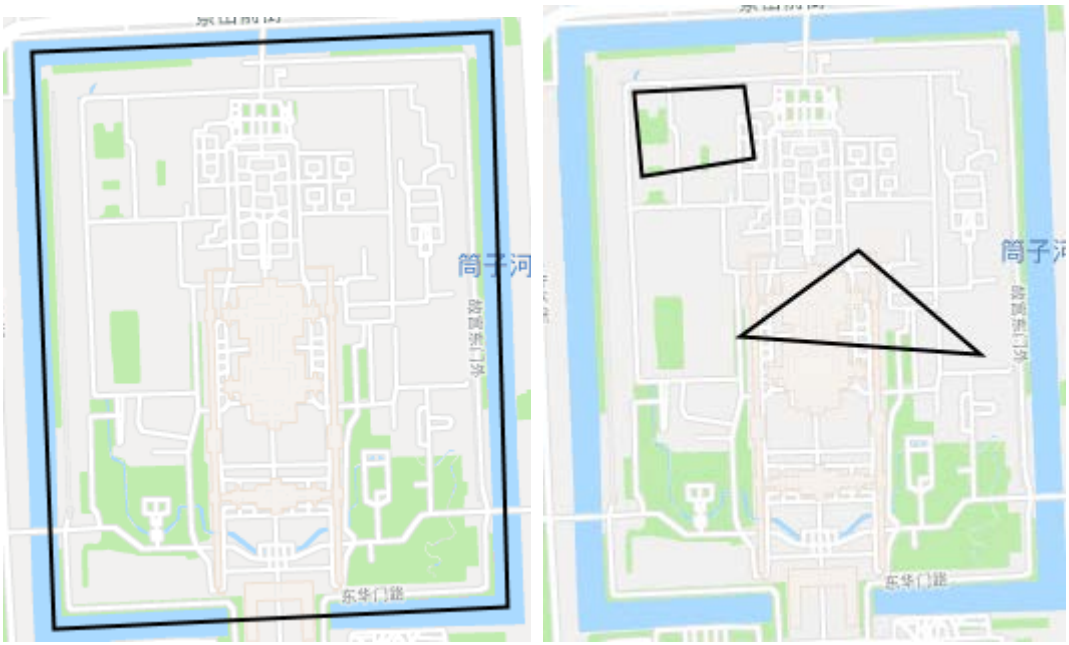


图 6.3 单线环和多线环的创建

下边介绍面和正方形的创建，代码及操作效果如下所示：

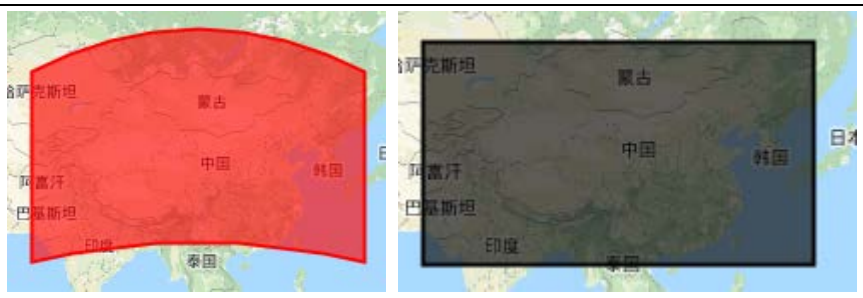
```
var Forbidden_City =ee.Geometry.Polygon([
                [[116.39179060756112, 39.922576684295926],
                 [116.39230559169198, 39.913064366936894],
                 [116.40213320552255, 39.91352520169099],
                 [116.40161822139169, 39.92297163084022]],

                [[116.39483759700204, 39.91961451259886],
                 [116.39496634303475, 39.91648763678902],
                 [116.39900038539315, 39.916553467224674],
                 [116.39874289332772, 39.91977908105491]]       ]);

Map.centerObject(Forbidden_City,15)
print(Forbidden_City)
Map.addLayer(Forbidden_City)
```

```
var Forbidden_City = ee.Geometry.Rectangle(
   116.3926, 39.9133, 116.4014, 39.9227);

Map.centerObject(Forbidden_City)
print(Forbidden_City)
Map.addLayer(Forbidden_City)
```

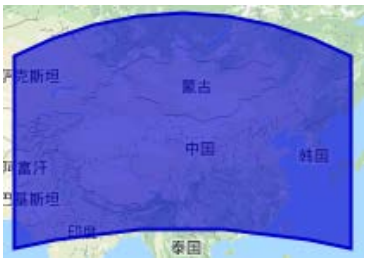
图 6.4 面和正方形的创建

下边介绍 Geometry 的坐标系转换与显示效果，代码和执行效果如下：

```
var China_Geo     = ee.Geometry.Rectangle(65.9, 19.8,134.5, 50.9);
var China_Planr = ee.Geometry(China_Geo, null, false)
var China_EPSG = China_Geo.transform('EPSG:3857', ee.ErrorMargin(100))

Map.addLayer(China_Geo,          {color: 'FF0000'},    'geodesic polygon')
Map.addLayer(China_Planr,        {color: '000000'},    'planar polygon')
Map.addLayer(China_EPSG,         {color: '0000CD'},    'EPSG polygon ')
```



China_Geo　　China_Planr



China_EPSG

图 6.5 坐标系转换与不同坐标系的显示效果

本例中.transform('EPSG:3857', ee.ErrorMargin(100))的功能是将原来 Geometry 的坐标系转换成指定坐标系，该命令中有两个参数，第一个参数是要转换的目标坐标系，第二个参数是转换后的允许误差。另外，ee.Geometry(China_Geo, null, false)的三个参数分别为“矢量文件”,“坐标系”和“显示方式”,默认情况下第三个参数是 true,代表用地理坐标系来显示文件。本例中此处改为了 false，这意味着将用投影坐标系来显示文件。

更进一步的讲，本例中 China_Geo 虽然是一个矩形，但是在默认的地理坐标系显示下矩形的上下边是弯曲的；China_Planr 与 China_Geo 是同样的数据，但由于将显示坐标系改成了投影投影坐标系的方式，所以显示出了正常的矩形；而 China_EPSG 是经过.transform()命令转换，并且转换为了投影坐标系的系统(EPSG3857:WGS 84/Pseudo-Mercator-Spherical Mercator 系统，详情参见 https://epsg.io)的，但在显示的时候仍然按照默认的地理坐标系进行显示。

下边介绍求取 Geometry 中心点的命令，代码及执行效果如下：

```
var China_Geo      = ee.Geometry.Rectangle(65.9, 19.8,134.5, 50.9);
var China_Certer   = China_Geo.centroid();

Map.addLayer(China_Geo,  {color: 'FF0000'}, 'geodesic polygon')
Map.addLayer(China_Certer)
```

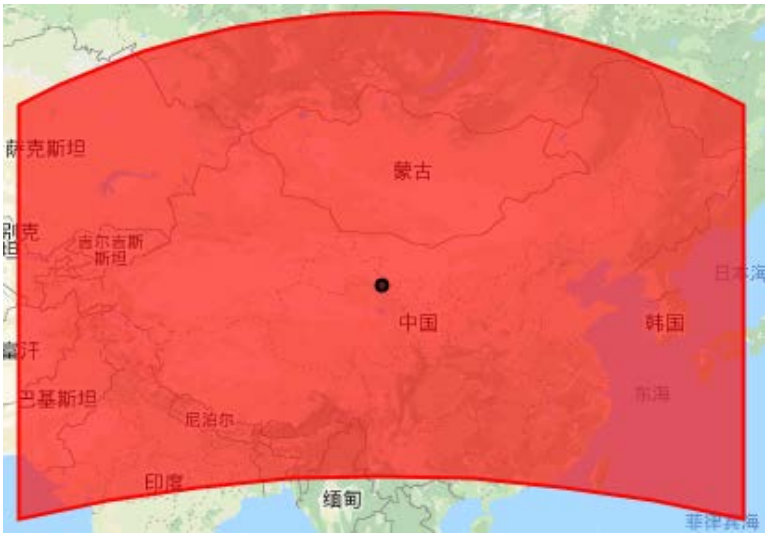


图 6.6 求取 Geometry 中心点

下边介绍求取 Geometry 的简化命令，代码及执行效果如下：

```
var TGR = ee.FeatureCollection("users/wangjinzhulala/TGR").geometry();
var TGR_Simple = TGR.simplify(50000)

Map.centerObject(TGR )
Map.addLayer(TGR)
Map.addLayer(TGR_Simple)
```

图 6.7 简化 Geometry

下边介绍求取 Geometry 四至和凸包的命令，代码及执行效果如下：

```
var TGR          =    ee.FeatureCollection("users/wangjinzhulala/TGR").geometry();
var TGR_Bound    =    TGR. bounds()
var TGR_Hall     =    TGR.convexHull()

Map.centerObject(TGR )
Map.addLayer(TGR)
Map.addLayer(TGR_Bound)
Map.addLayer(TGR_Hall)
```

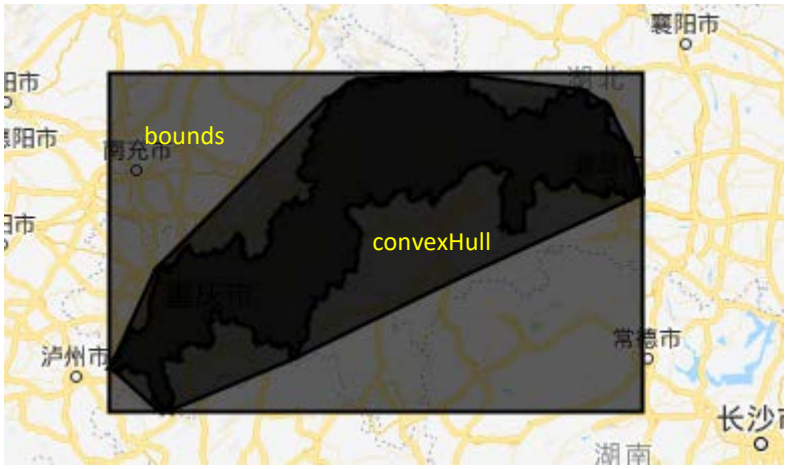


图 6.8 求取 Geometry 四至和凸包

下边介绍求取 Geometry 缓冲区的命令，代码及执行效果如下：

```
var TGR          = ee.FeatureCollection("users/wangjinzhulala/TGR").geometry();
var TGR_Buffer = TGR.buffer(10000)

Map.centerObject(TGR )
Map.addLayer(TGR)
Map.addLayer(TGR_Buffer)
```

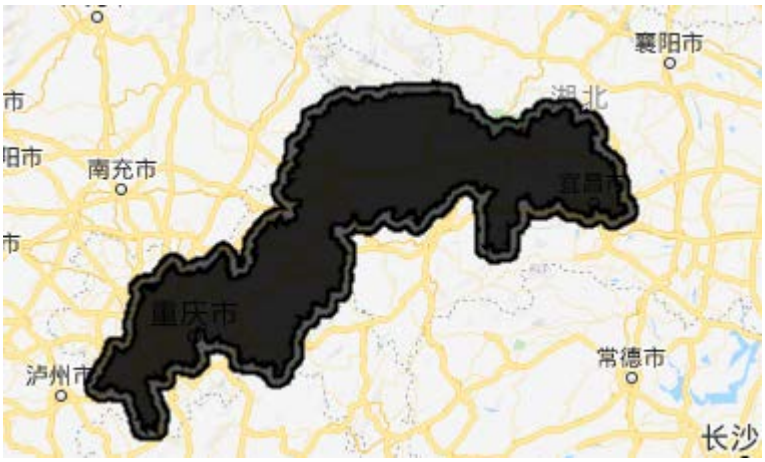


图 6.9 求取 Geometry 的缓冲区

下边介绍求取 Geometry 的并集操作，代码及执行效果如下：

```
var Polygon_1 = ee.Geometry.Polygon(
        [[[152.71032334056088, 18.862610207551867],
          [-170.72717665943918, 19.69222816299581],
          [-171.07873915943918, 40.62061002589569],
          [153.41344834056088, 40.08478212000696]]]),
    Polygon_2 = ee.Geometry.Polygon(
        [[[-176.52795790943912, 45.620172046005735],
          [-176.70373915943912, 32.2945468815855],
          [-142.60217665943918, 32.2945468815855],
          [-141.02014540943918, 45.49709377875777]]]);

var Polygon_union = Polygon_1.union(Polygon_2);

Map.centerObject(Polygon_union)
Map.addLayer(Polygon_1)
Map.addLayer(Polygon_2)
Map.addLayer(Polygon_union)
```

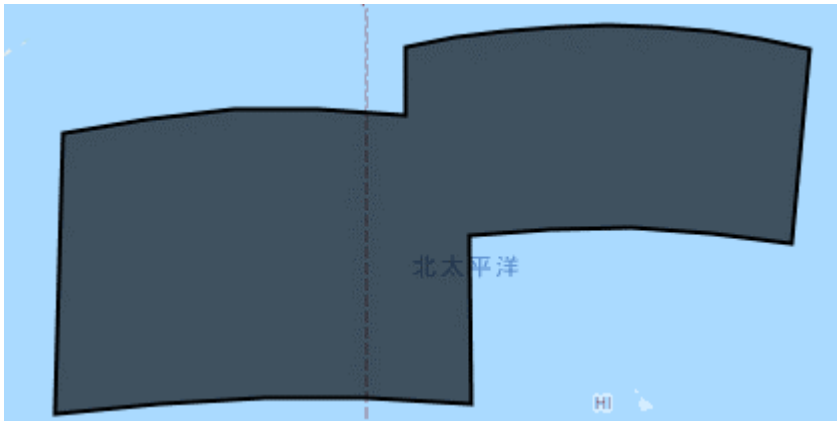


图 6.10 求取 Geometry 的并集

需要注意，GEE 中其它针对 Geometry 的空间关系操作如下：

.intersection()	.symmetricDifference()	.Difference()
相交	对称求反	求反

下边介绍求 Geometry 的要素分解命令，代码如下：

```
var Forbidden_City = ee.Geometry.MultiPolygon(
        [[[[116.39393637477303, 39.921260179361866],
           [116.39350722133065, 39.91882457850439],
           [116.39676878749276, 39.918791664386056],
           [116.39711211024667, 39.920963962263336]]],

         [[[116.39887163936044, 39.91810046424626],
           [116.39595339595223, 39.91698136351405],
           [116.40080282985116, 39.91652055201477]]]]);

var Geometry_List = Forbidden_City.geometries()
Map.centerObject(Forbidden_City)
Map.addLayer(Forbidden_City)

print(Geometry_List)
```

本例中，Forbidden_City 由一个四边形和一个三角形组成，因此.geometries()可以将这个 Geometry 集合展开为单独的形状要素，效果如下：

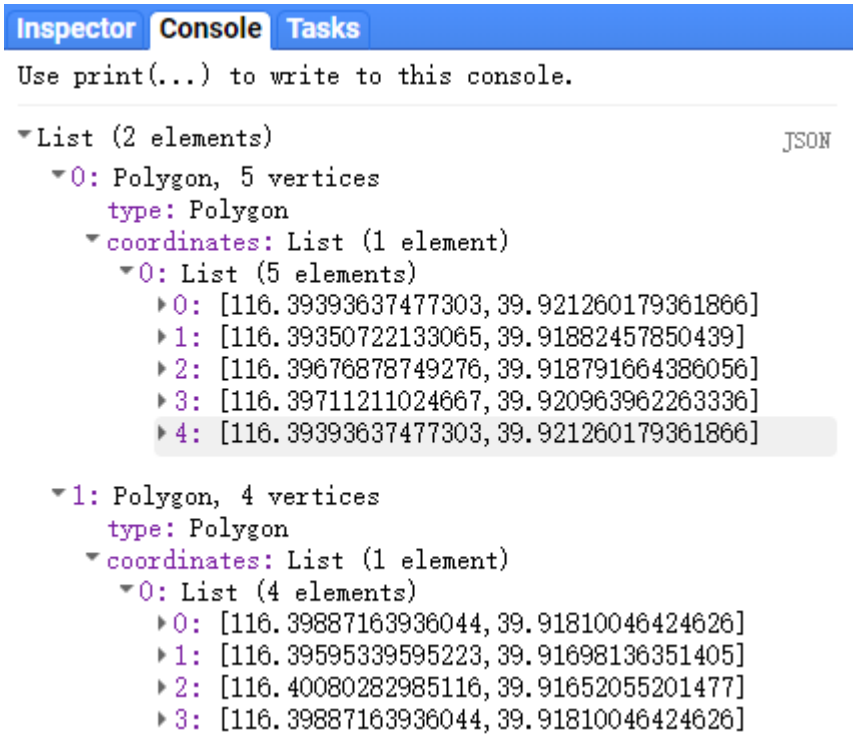


图 6.10 Geometry 的要素分解

下边介绍求 Geometry 的面积和周长求取命令，代码和执行效果如下：

```
var Polygon_1 =ee.Geometry.Polygon([
                [[116.39179060756112, 39.922576684295926],
                 [116.39230559169198, 39.913064366936894],
                 [116.40213320552255, 39.91352520169099],
                 [116.40161822139169, 39.92297163084022]],

                [[116.39483759700204, 39.91961451259886],
                 [116.39496634303475, 39.91648763678902],
                 [116.39900038539315, 39.916553467224674],
                 [116.39874289332772, 39.91977908105491]]       ]);
var Polygon_2 =ee.Geometry.Polygon(
              [[[116.39239142238046, 39.913146659084674],
                [116.40209029017831, 39.91354166001771],
                [116.40170405208016, 39.922988086896794],
                [116.39183352290536, 39.922593140447425]]]);

var Area_1 = Polygon_1.area()
var Area_2 = Polygon_2.area()
var Perimeter_1 = Polygon_1.perimeter()
var Perimeter_2 = Polygon_2.perimeter()

print(Area_1,Area_2,Perimeter_1,Perimeter_2)
Map.centerObject(Polygon_1,15)
Map.addLayer(Polygon_1)
Map.addLayer(Polygon_2)
```

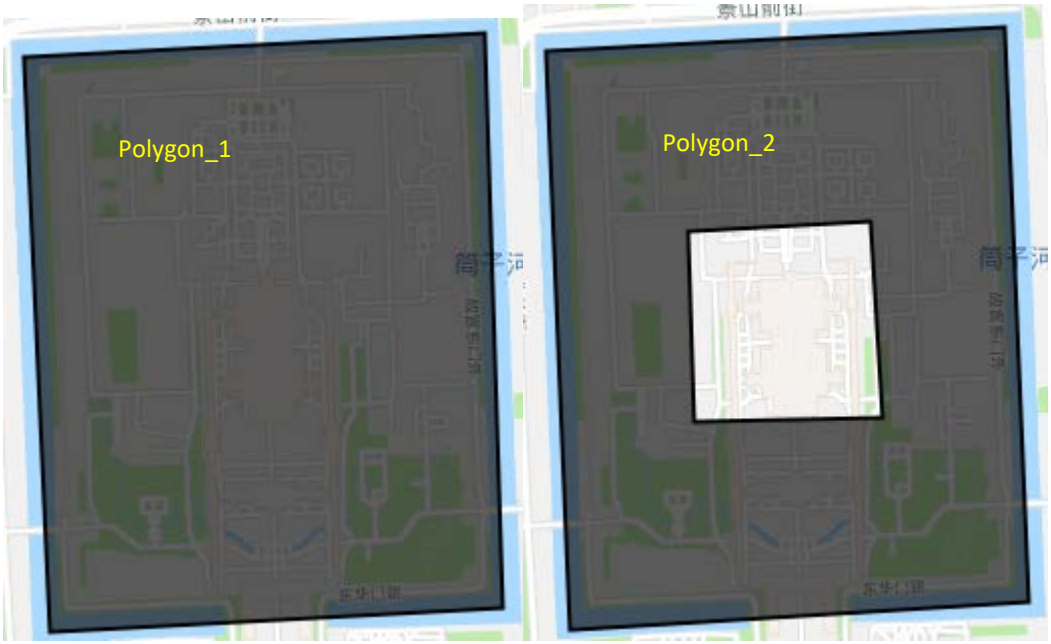


图 6.11 Geometry 面积和周长的求取

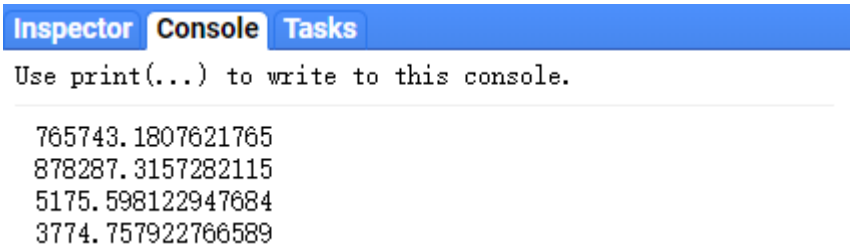


图 6.12 Geometry 面积和周长的求取结果

下边介绍求 Geometry 间的最短距离计算，代码和执行效果如下：

```
var geometry_1 = ee.Geometry.Polygon(
        [[[116.39419386683858, 39.92130954875369],
          [116.39359305201924, 39.918873949652266],
          [116.3966829568044, 39.918709379020974],
          [116.39719794093526, 39.92101333186874]]]),
    geometry_2 = ee.Geometry.Polygon(
        [[[116.39891455470479, 39.918116921473526],
          [116.3957817345754, 39.917063650955505],
          [116.40123198329366, 39.91647117917021]]]);
var Distance = geometry_1.distance(geometry_2)

print (Distance)

Map.centerObject(geometry_1)
Map.addLayer(geometry_1)
Map.addLayer(geometry_2)
```

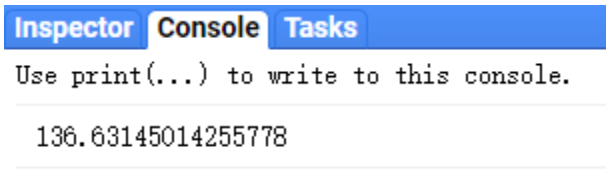


图 6.13 Geometry 间的最短距离

本节学习的 Geometry 常见命令如下，尝试回想其语法和功能：

ee.Geometry.Point()	ee.Geometry.Multipoint()	Geometry.transform()
geometry.centroid ()	geometry.simplify ()	geometry.bounds()
geometry.buffer ()	geometry.union ()	geometry.geometries()
geometry.length ()	geometry.area ()	geometry.perimeter ()
geometry.distance ()		

6.2 Feature

Feature 是“带有属性的 Geometry”，因此对其除了能运用上节中的空间操作外，还可以根据其属性进行相关操作。

下边介绍 Feature 的创建，代码及操作效果如下：

```
var Forbidden_City =ee.Geometry.Polygon(
        [[ [116.392391422, 39.9131466590], [116.402090290, 39.913541660],
          [116.401704052, 39.9229880868], [116.391833522, 39.922593140]]]);
var Feature_Fobidden_City = ee.Feature(
  Forbidden_City,
  {name:'故宫', location:'北京'})

Map.centerObject(Forbidden_City)
print(Feature_Fobidden_City)
Map.addLayer(Feature_Fobidden_City)
```

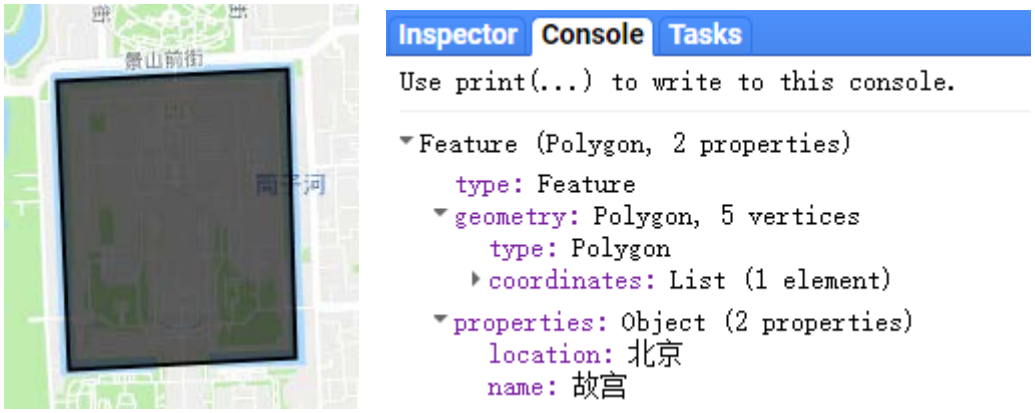


图 6.11 Feature 的创建

在这里，ee.Feature( )一共有两个参数，第一个是 Feature 的 Geometry 数据，另一个是 Feature 的属性信息，属性信息要用 Dictionary 的结构来写入。值得一提的是，利用代码方式创建 Feature 的效率很低，因此与 Geometry 一样，我们可以通过“手绘工具”交互式的创建 Feature 并为其录入信息以提高效率。

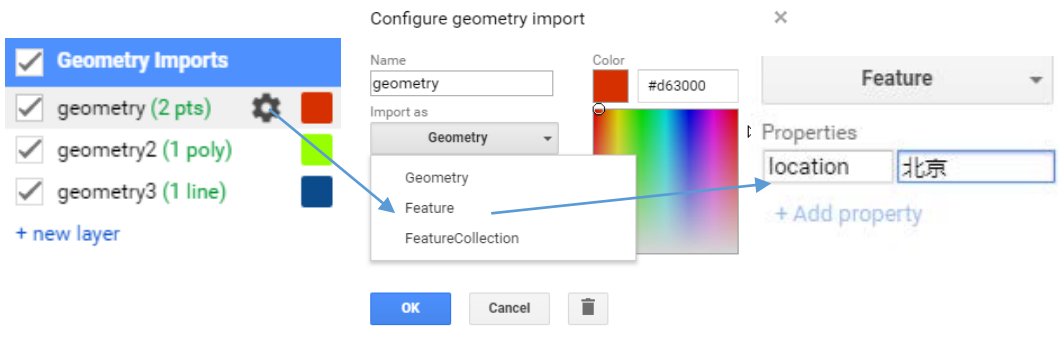


图 6.12 Feature 的手绘创建

下边介绍 Feature 的选择创建，代码及操作效果如下：

```
var Forbidden_City =ee.Geometry.Polygon(
        [[[116.39239142238046, 39.913146659084674],
          [116.40209029017831, 39.91354166001771],
          [116.40170405208016, 39.922988086896794],
          [116.39183352290536, 39.922593140447425]]]);
var Feature_Fobidden_City = ee.Feature(
   Forbidden_City,
   {name:'故宫', location:'北京'})
var Feature_select_1 = Feature_Fobidden_City.select(['name'])
var Feature_select_2 = Feature_Fobidden_City.select(['name'],['名称'])



Map.centerObject(Forbidden_City)
print(Feature_Fobidden_City, Feature_select_1,Feature_select_2)
Map.addLayer(Feature_Fobidden_City)
```

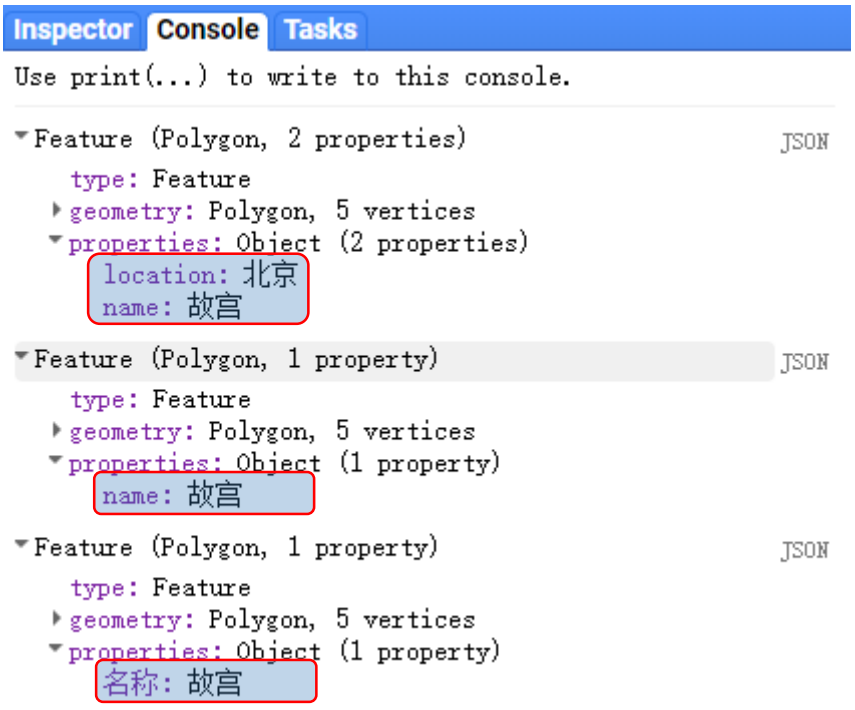


图 6.13 Feature 的选择创建

Feature.**select()**命令的功能是创建一个原 Feature 的“拷贝”，当括号内只有一个参数时，GEE 拷贝原 Feature 的 Geometary，并将参数对应的属性加入到这个 Geometry 中形成新的 Feature；当括号内有两个参数时，GEE 在将参数加入 Geometry 的基础上，利用第二个参数对属性的 Key 进行重命名。应该注意的是，这个命令要求所有参数必须是 List 的形式，即参数两端需要由方括号引导，并且参数的形式为文本。

下边介绍 Feature 的属性改写，代码及操作效果如下：

```
var Forbidden_City          = ee.Geometry.Rectangle( 116.3926, 39.9133, 116.4014, 39.9227);
var Feature_Fobidden_City = ee.Feature(Forbidden_City, {name:'故宫', location:'北京'})
var Feature_Set             = Feature_Fobidden_City.set('name','Gugong', 'location','Beijing')
var Feature_Set_Multi       = Feature_Fobidden_City.setMulti({'name':'紫禁城 ', 'location':'首都 '})

print(Feature_Fobidden_City,Feature_Set,Feature_Set_Multi)
Map.centerObject(Forbidden_City)
Map.addLayer(Feature_Fobidden_City)
```

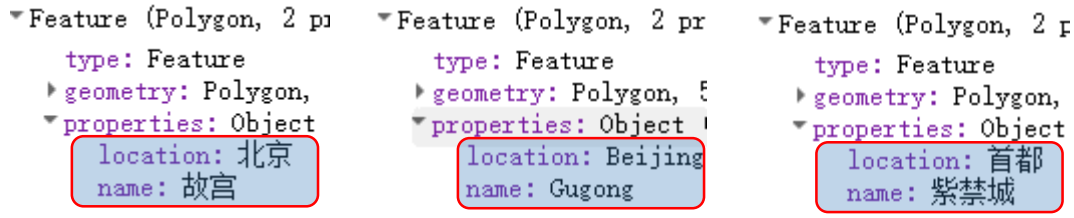


图 6.14 Feature 的属性改写

下边介绍 Feature 的空间操作，代码及操作效果如下：

```
var China          = ee.FeatureCollection("users/wangjinzhulala/China_Provinces");
var Chongqing      = ee.Feature(China.filterBounds(ee.Geometry.Point([106,29])).first())
var CQ_Simple      = Chongqing.simplify(50000)
var CQ_Centroid    = Chongqing.centroid()
var CQ_Hull        = Chongqing.convexHull()
var CQ_Bounds      = Chongqing.bounds()
var CQ_Buffer      = Chongqing.buffer(50000)
Map.centerObject(Chongqing)
Map.addLayer(Chongqing)
Map.addLayer(CQ_Simple)
Map.addLayer(CQ_Centroid)
Map.addLayer(CQ_Hull)
Map.addLayer(CQ_Bounds)
Map.addLayer(CQ_Buffer)
```

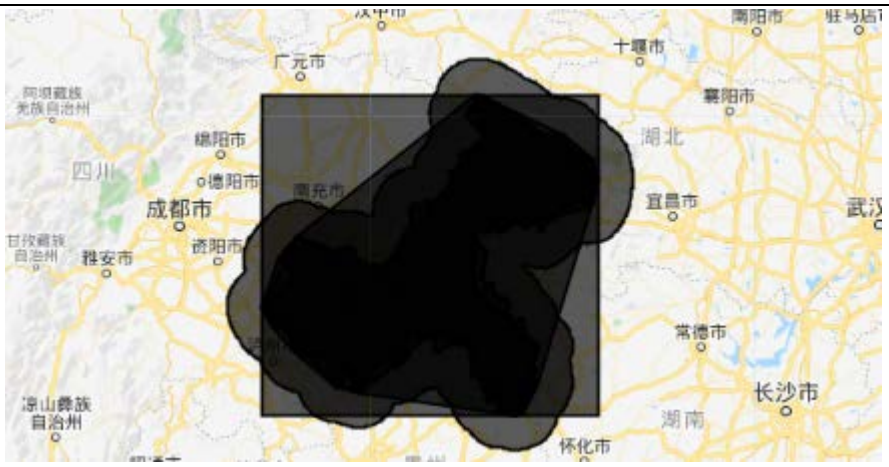


图 6.14 Feature 的空间操作

下边介绍 Feature 的交并操作，代码及操作效果如下：

```
var China = ee.FeatureCollection("users/wangjinzhulala/China_Provinces");
var Chongqing = ee.Feature(China.filterBounds(ee.Geometry.Point([106,29])).first())
var TGR = ee.Feature(ee.FeatureCollection("users/wangjinzhulala/TGR").first());

var Union = Chongqing.union(TGR)
var Intersect = Chongqing.intersection(TGR)
var Difference = Chongqing.difference(TGR)
var Symmetric_Difference = Chongqing.symmetricDifference(TGR)

Map.centerObject(Chongqing)
Map.addLayer(Union,{color:'00ffff'},'Union')
Map.addLayer(Intersect,{color:'DC143C'},'Intersect')
Map.addLayer(Difference,{color:'7B68EE'},'Difference')
Map.addLayer(,{color:'32CD32'},'Symmetric_Difference')
```

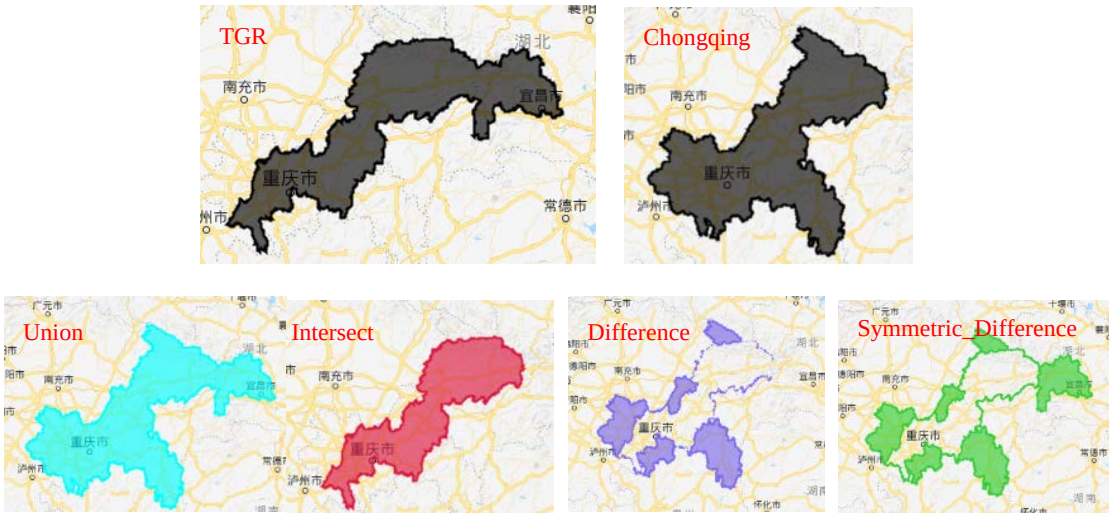


图 6.15 Feature 的交并操作

本例中运用到了.filterBound()和.first()，它们的功能是利用地理位置进行筛选和取第一个数据。这些命令会在后边进行介绍，这里可以先对其有概念上的认识。

下边介绍 Feature 的空间信息和属性信息的提取，代码如下：

```
var TGR = ee.Feature(ee.FeatureCollection("users/wangjinzhulala/TGR").first());
var TGR_geometry = TGR.geometry();
var TGR_get = TGR.get('BOUNT_ID')

print(TGR,TGR_geometry,TGR_get)
```

本例中，.geometry()的功能相当于提取 Feature 的空间信息，而 get('BOUNT_ID')的功能是获取 Feature 属性中相应 Key 的内容信息，该代码是执行效果如下：

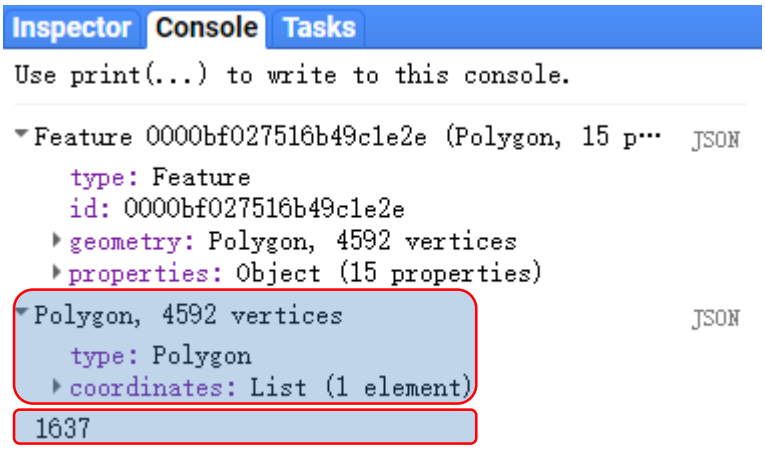


图 6.16 Feature 的空间信息和属性信息的提取

下边介绍 Feature 的面积和周长计算，代码如下：

```
var TGR = ee.Feature(ee.FeatureCollection("users/wangjinzhulala/TGR").first());
var TGR_Area = TGR.area()
var TGR_Perimeter = TGR.perimeter()

print(TGR_Area,TGR_Perimeter)
```

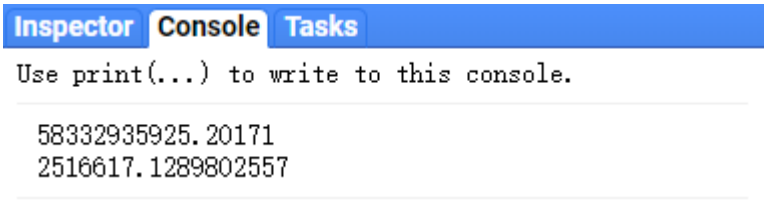


图 6.17 Feature 的面积和周长计算

可以看出，Geometry 与 Feature 在空间操作上的命令形式是一样的，可以统一的记忆这些命令以提高学习效率。

下边是本节涉及到的有关 Feature 的常见命令，尝试回忆其格式与功能：

ee.Feature()	Feature.select()	Feature.transform()	Feature.set/setMulti()
Feature.geometry()	Feature.get()	Feature.Length/Area/Perimeter()	
Feature.centroid/simplify/bounds/convexHull/buffer()			
Feature.union/intersection/difference/symmetricDifference()			

6.3 Feature Collection

Feature Collection 可以看做是一个或多个 Feature 文件的集合。类比 Feature 数据比 Geometry 数据多出了属性信息，Feature Collection 相对 Feature 也多出了一定的“数据集”信息。比如，中国各省级行政边界都可以看做是的 Feature，都包含有各自的“Name”、“Area”

等信息，而当将这些 Feature 集合到一起时，就形成了一个“中国省边界”的 Feature Collection，这个 Collection 中可以包含“数据来源”、“采集时间”等属于数据集合的信息。

下边介绍 GEE 自带的 Feature Collection 数据(图 6.18)。在搜索栏中输入“Table”之后观察弹出框的 TABLES 部分，点击 import 按钮即可将对应数据加载到代码中。

图 6.18 GEE 自带的 Feature Collection 数据

对于 GEE 找不到的数据，也可以通过上传的方式形成用户的个人数据。通常可以利用 ArcGIS 来进行 Esri Shapefile (SHP)矢量格式文件的上传。具体操作如下图所示：

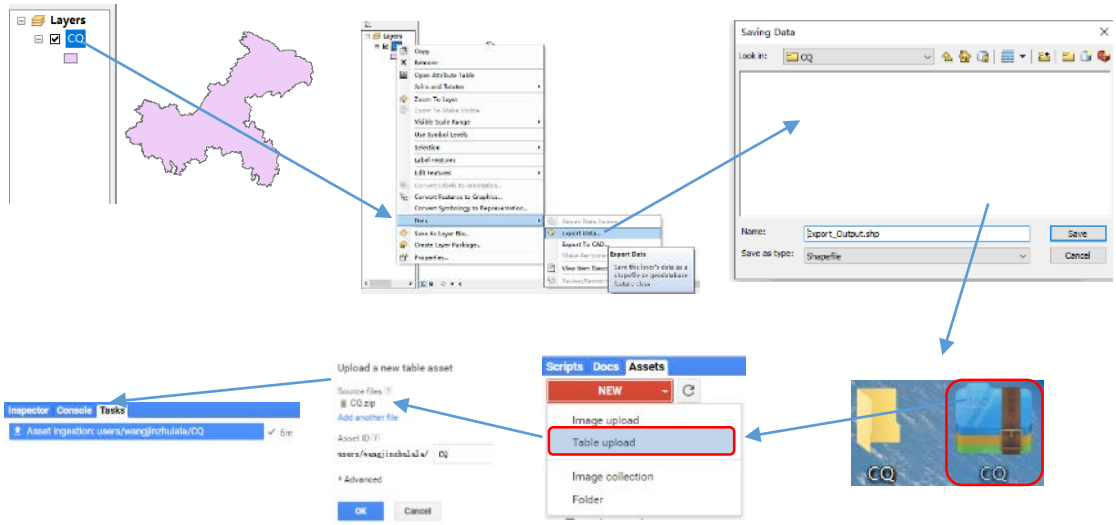


图 6.19 上传用户个人数据

其它引入 Feature Collection 的方式是利用谷歌公司的 Fusion Table 代码进行加载。值得一提的是，早期 GEE 不支持 SHP 矢量文件的直接上传，因此 Fusion Table 作为一种间接方式(用户先把 SHP 上传到 Fusion Table，再利用 ID 进行加载)得到了一定的应用，而目前谷歌公司已经决定在 2019 年 12 月 3 日的时候停止 Fusion Table 服务。因此，建议 GEE 的学习者采用 Table 上传的方式处理个人数据。

下边介绍 Feature Collection 的创建。因为 Feature Collection 是由一系列 Feature 数据组成的，因此其创建的前提是首先拥有 Feature 数据。创建 Feature 数据后，将这些数据装在一个“数据篮”(List)中，然后利用命令告诉告 GEE 这个篮子里都是 Feature 数据，就能生成相应的 Feature Collection 数据集。上述思路的代码及执行效果如下：

```
var Poly     = ee.Geometry.Polygon(
            [[[116.39248531478768, 39.91319513792091],
              [116.40222709792977, 39.91378763804685],
              [116.40184085983162, 39.92303655876662],
              [116.39171283859139, 39.922674524864675]]]);
var Points = ee.Geometry.MultiPoint(
            [[116.38823669570809, 39.92079850034943],
             [116.38677757400399, 39.91885659627606],
             [116.38669174331551, 39.916881722097074],
             [116.38767879623299, 39.91421555161684]]);
var Lines    = ee.Geometry.LineString(
            [[116.37733619827156, 39.9167171466781],
             [116.38192814010506, 39.91754001981801],
             [116.37712162155037, 39.91365597179514],
             [116.38424556869393, 39.914478881715695]]);

var Feature_Collection = ee.FeatureCollection( [Poly, Points, Lines] );

Map.centerObject(Feature_Collection)
print (Feature_Collection)
Map.addLayer (Feature_Collection)
```

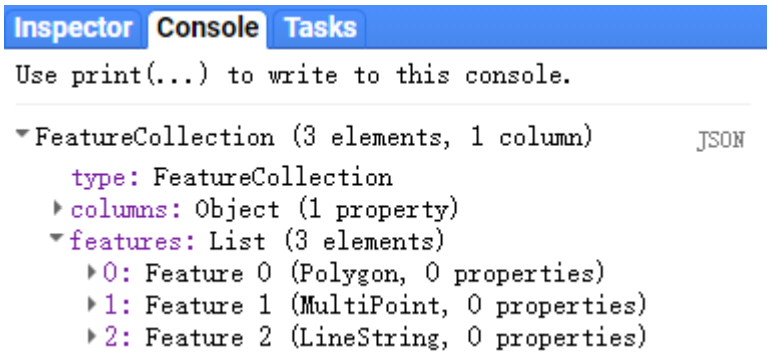


图 6.20 Feature Collection 的创建

下边介绍 Feature Collection 的随机点生成方式，代码及执行效果如下：

```
var Poly     = ee.Geometry.Polygon(
            [[[116.39248531478768, 39.91319513792091],
              [116.40222709792977, 39.91378763804685],
              [116.40184085983162, 39.92303655876662],
              [116.39171283859139, 39.922674524864675]]]);
var Random_Points = ee.FeatureCollection.randomPoints(Poly,20)

Map.centerObject(Poly)
Map.addLayer(Poly)
Map.addLayer(Random_Points)
```

图 6.21 Feature Collection 的随机点生成

下边介绍 Feature Collection 的属性筛选，代码及执行效果如下：

```
var Proction_Area = ee.FeatureCollection("WCMC/WDPA/current/polygons");
var Proction_Area_China = Proction_Area.filterMetadata('ISO3','equals','CHN')

Map.centerObject(Proction_Area_China)
Map.addLayer(Proction_Area )
Map.addLayer(Proction_Area_China ,{color:'FF0000'})
```

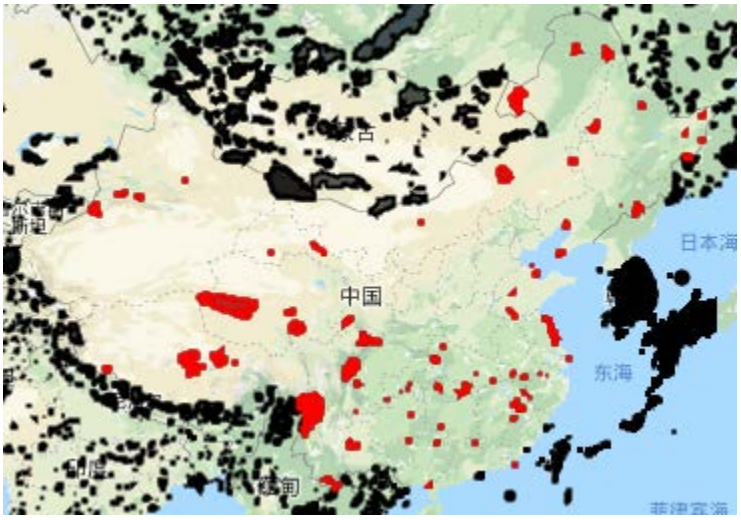


图 6.21 Feature Collection 的属性筛选

这里需要注意两点。第一是加载的数据集是“世界自然保护区”，其中包含有很多属性信息，“ISO3”指的是 3 字母国家代码。第二是.filterMetadata()的功能是利用属性进行筛选，其三个参数分别是“筛选字段”，“关系”和“筛选值”，其中“关系”字段包括"equals", "less_than",

"greater_than","not_equals","not_less_than","not_greater_than","starts_with","ends_with","not_starts_with", "not_ends_with", "contains","not_contains"。

下边介绍 Feature Collection 的数量限制命令，代码及执行效果如下：

```
var China_Provinces = ee.FeatureCollection("users/wangjinzhulala/China_Provinces");

Map.addLayer(China_Provinces)
Map.addLayer(China_Provinces.limit(5,'Shape_Area',false))
```

图 6.22 Feature Collection 的数量限制

数量限制命令.limit()有三个参数，分别是“最大数量”，“限制字段”和“是否从小到大”。其中“限制字段”和“是否从小到大”是非必须参数。数量限制命令常用来观察数据，因为 GEE 存在某些 Feature 数量很多的 Feature Collection 数据，而 GEE 只支持将一定数量的数据显示在 Console 栏中，此时可以该命令来加载部分数据进而方便对数据进行观察。

下边介绍时间筛选命令，由于带有时间属性的 Feature 数据很少，这里用 Image Collection 代替 Feature Collection，代码及执行效果如下：

```
var L8 = ee.ImageCollection("LANDSAT/LC08/C01/T1_RT");
var Landsat_FilterDate = L8.filterDate('2019-03-01','2019-03-16').limit(50)

print(Landsat_FilterDate)
```

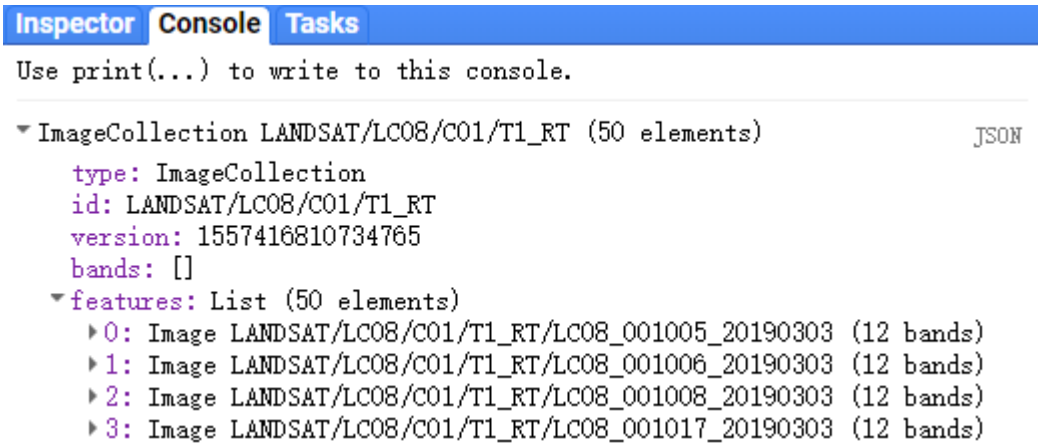


图 6.22 时间筛选命令

下边介绍 Feature Collection 的空间筛选命令，代码及执行效果如下：

```
var China_Provinces = ee.FeatureCollection("users/wangjinzhulala/China_Provinces");
var CQ_Point =ee.Geometry.Point([106.92371845031437, 29.430312117372274]);
var CQ = China_Provinces.filterBounds(CQ_Point)

Map.centerObject(China_Provinces,3)
Map.addLayer(China_Provinces)
Map.addLayer(CQ,{color:'FF0000'})
Map.addLayer(CQ_Point)
```

图 6.23 空间筛选命令

下边介绍 Feature Collection 的选择复制命令，代码及执行效果如下：

```
var China_Provinces = ee.FeatureCollection("users/wangjinzhulala/China_Provinces");
var China_Area = China_Provinces.select(['NAME','Shape_Area'])

print(China_Provinces)
print(China_Area)

Map.centerObject(China_Provinces,4)
Map.addLayer(China_Provinces)
Map.addLayer(China_Area)
```

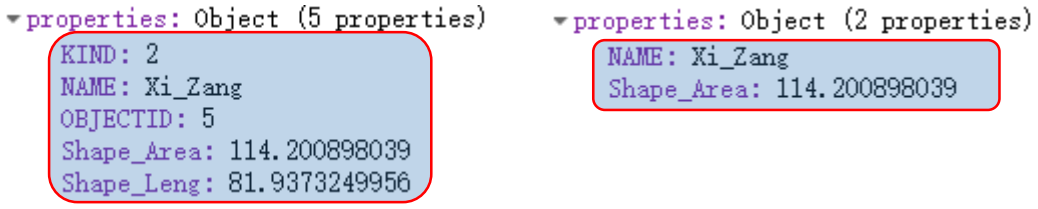


图 6.24 Feature Collection 的选择复制

下边介绍 Feature Collection 的去除重复命令，代码及执行效果如下：

```
var China_Provinces = ee.FeatureCollection("users/wangjinzhulala/China_Provinces");
var China_Area = China_Provinces.distinct(['NAME'])

print(China_Provinces)
print(China_Area)

Map.centerObject(China_Provinces,4)
Map.addLayer(China_Provinces)
Map.addLayer(China_Area)
```

OBJECTID *	Shape *	NAME	
36	Polygon	An_Hui	2
35	Polygon	Ao_Men	2
38	Polygon	Chong_Qing	2
33	Polygon	Fu_Jian	2
34	Polygon	Fu_Jian	2
32	Polygon	Gan_Su	2
31	Polygon	Guang_Dong	2
1089	Polygon	Guang_Dong	2
30	Polygon	Guang_Xi	2
29	Polygon	Gui_Zhou	2

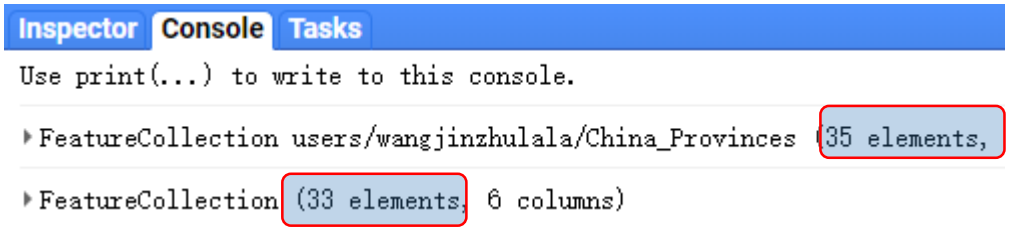


图 6.25 Feature Collection 的去重命令

去除重复命令.distinct()的作用是去除具有相同属性 Feature。本例中，因为福建省和广东省附近的小岛具有与陆地边界一样的 NAME，所以当执行.distinct()时，这些小岛就被去除了，因此可以看出 Feature Collection 有原来的 35 个元素变成了 33 个。

下边介绍 Feature Collection 的联合和融合命令，联合的效果是形成一个单独的 Feature，并且抹去所有属性，融合是将两个 Collection 合并为一个，代码及执行效果如下：

联合能够将多个 Feature 进行处理，其代码如下：

```
var China_Provinces = ee.FeatureCollection("users/wangjinzhulala/China_Provinces");
var China_Union = China_Provinces.union()

print(China_Provinces)
print(China_Union)

Map.centerObject(China_Provinces,4)
Map.addLayer(China_Provinces)
Map.addLayer(China_Union,{color:'ff0000'})
```

融合只能对两个 FeatureCollection 进行处理，其代码如下：

```
var China_Provinces = ee.FeatureCollection("users/wangjinzhulala/China_Provinces");
var Chongqing = China_Provinces.filterMetadata('NAME','equals','Chong_Qing')
var Si_Chuan = China_Provinces.filterMetadata('NAME','equals','Si_Chuan')
var CQ_SC_Merge = Chongqing.merge(Si_Chuan)

Map.centerObject(China_Provinces,4)
Map.addLayer(Chongqing)
Map.addLayer(Si_Chuan,{color:'0000ff'})
Map.addLayer(CQ_SC_Merge,{color:'ff0000'})
```

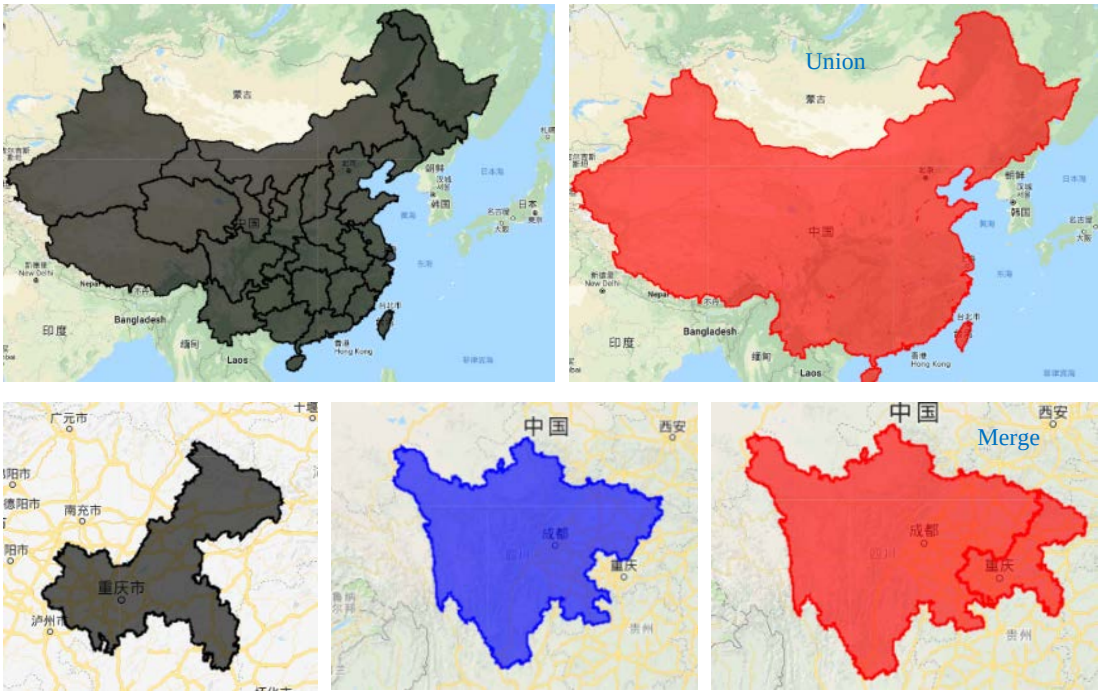


图 6.26 Feature Collection 的联合和融合

下边介绍 Feature Collection 的属性改写，代码及执行效果如下：

```
var China_Provinces = ee.FeatureCollection("users/wangjinzhulala/China_Provinces");
var Chongqing = China_Provinces.filterMetadata('NAME','equals','Chong_Qing')
var Si_Chuan = China_Provinces.filterMetadata('NAME','equals','Si_Chuan')


var CQ_SC_Merge = Chongqing.merge(Si_Chuan)
var CQ_SC_Merge_Set = CQ_SC_Merge.set('NAME','Chuan_Yu')


print(CQ_SC_Merge)
print(CQ_SC_Merge_Set)
```

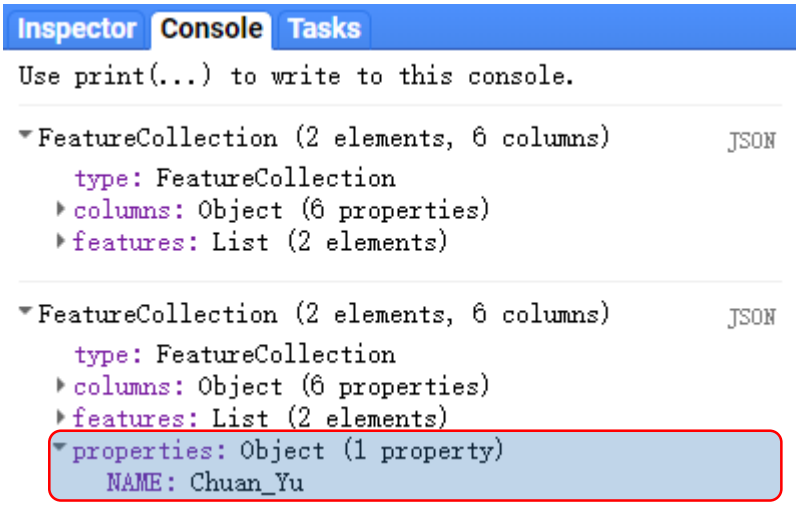


图 6.27 Feature Collection 的属性改写

本例中通过 .set() 命了增加了属于 Feature Collection 的属性信息，同理也可以通过 .setMulti() 来改写属性，具体操作可以参考前述内容。

下边介绍 Feature Collection 的 Remap 命令，代码及执行效果如下：

```
var China_Provinces = ee.FeatureCollection("users/wangjinzhulala/China_Provinces");
var Old_Provinces_ID = ee.List([1,2,3,4,5,6,7,8,9,12,13,14,15,16,18,19,20,21,
                                22,23,24,27,28,29,30,31,32,33,34,35,36,37,38,39,1089])
var New_Provinces_ID = ee.List([1,1,1,1,2,2,2,2,2,3,3,3,3,3,4,4,4,4,4,4,5,5,5,5,5,6,6,6,6,6,7,7,7,7,7])

var China_Remap = China_Provinces.remap(Old_Provinces_ID,New_Provinces_ID,'OBJECTID')
var China_Provinces_Map = China_Provinces.reduceToImage(['OBJECTID'], ee.Reducer.first( ) )
var China_Remap_Map = China_Provinces.reduceToImage(['OBJECTID'], ee.Reducer.first( ) )

Map.centerObject(China_Provinces,4)
Map.addLayer(China_Provinces_Map,
             {min:1, max:40, palette:'16ff07,2901ff'},
             'China_Provinces_Map')
Map.addLayer(China_Remap_Map,
             {min:1, max:7, palette:'ff7248,fbff21,09ffe8'},
             'China_Remap_Map')
```

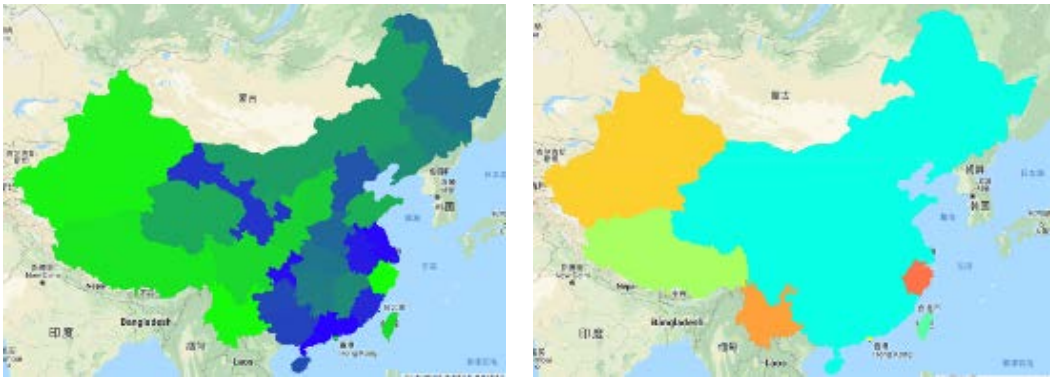


图 6.28 Feature Collection 的 Remap

可以把 remap.()命令理解为对 Feature 进行“批量修改”。该命令有三个参数，分别是“修改前的值”，“修改后的值”和“要修改的 Key”。本例的效果相当于将中国的省边界重新分为了 7 类，代码中.reduceToImage()的功能是“矢量转栅格”，用在这里的目的是突出显示效果，该命令会在后边进行介绍。

下边介绍 Feature Collection 的排序命令，代码及执行效果如下：

```
var China_Provinces = ee.FeatureCollection("users/wangjinzhulala/China_Provinces");
var China_Sort_Area = China_Provinces.sort('Shape_Area',false)

Map.centerObject(China_Provinces,4)
Map.addLayer(China_Sort_Area.limit(5),{color:'ff000000'})
```

图 6.29 Feature Collection 的排序

排序命令.sort()共有两个参数，一个是“排序字段”，另一个是“是否从小到大”。本例中利用面积进行排序，并且排序方式为 false(从大到小)，结合.limit()命令就筛选出了中国面积最大的前五个省。

下边介绍 Feature Collection 的 makeArray 命令，代码及执行效果如下：

```
var China_Provinces = ee.FeatureCollection("users/wangjinzhulala/China_Provinces").limit(3);
var China_Array =China_Provinces.makeArray(['OBJECTID','Shape_Area','Shape_Leng'],'An_Array')

print(China_Provinces)
print(China_Array)
```

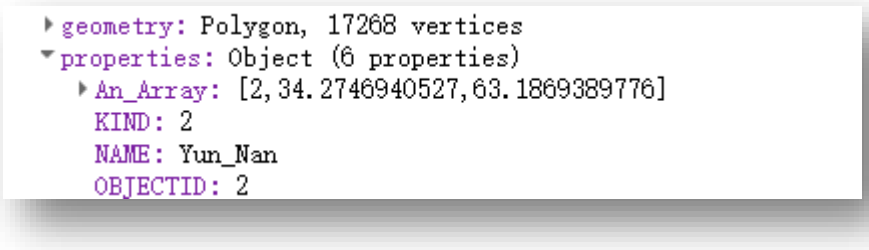


图 6.30 Feature Collection 的 makeArray 命令

下边介绍 Feature Collection 的提取 Geometry 命令，代码及执行效果如下：

```
var China_Provinces = ee.FeatureCollection("users/wangjinzhulala/China_Provinces").limit(3);
var China_Geometry = China_Provinces.geometry()

print(China_Provinces)
print(China_Geometry)
```

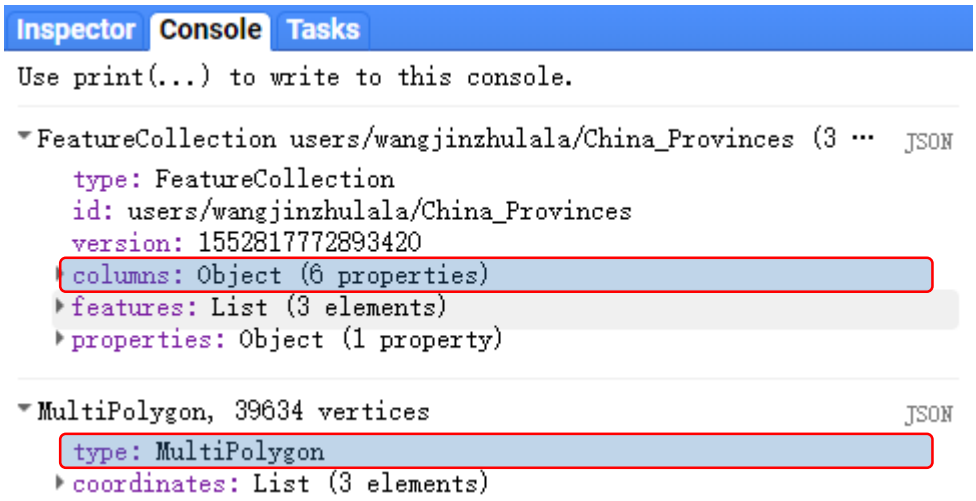


图 6.31 提取 Feature Collection 的 Geometry

下边介绍 Feature Collection 的矢量转栅格命令，代码及执行效果如下：

```
var China_Provinces = ee.FeatureCollection("users/wangjinzhulala/China_Provinces")
var China_to_Image = China_Provinces.reduceToImage(['OBJECTID'],ee.Reducer.first())

Map.centerObject(China_Provinces,4)
Map.addLayer(China_to_Image,{"min":1,"max":40,"palette":["ff9c07","f0ff1b","1aff0b"]})
```

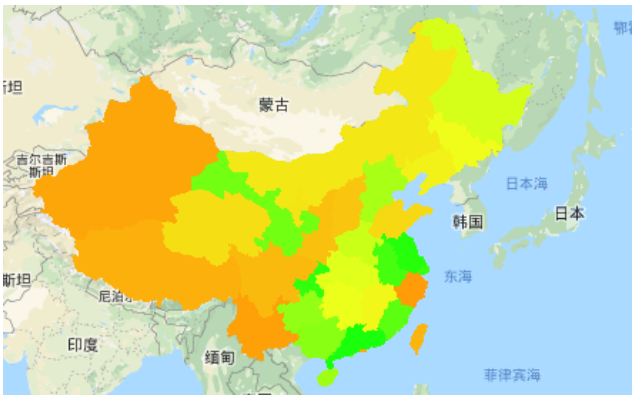


图 6.32 提取 Feature Collection 的矢量转栅格

矢量转栅格.reduceToImage()共有两个参数，第一个是“依据字段”，第二个是“数值处理”。本例中我们利用 ee.Reducer.first()的目的是不对数值进行任何处理。

下边介绍 Feature Collection 的取首个数据命令，代码及执行效果如下：

```
var China_Provinces = ee.FeatureCollection("users/wangjinzhulala/China_Provinces")
var China_Biggest_Province = ee.Feature(China_Provinces.sort('Shape_Area',false).first())
print(China_Biggest_Province)

Map.centerObject(China_Provinces,4)
Map.addLayer(China_Biggest_Province)
```

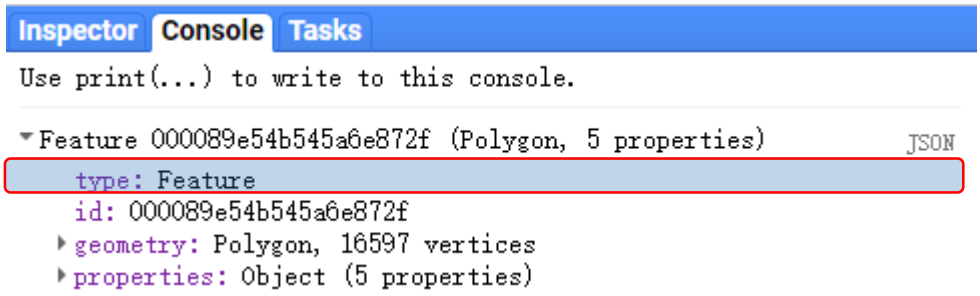


图 6.33 Feature Collection 的取首个数据命令

取首个.first()命令的常见目的是观察 Feature Collection 中数据的属性或者结构。

下边介绍 Feature Collection 转 List 命令，代码及执行效果如下：

```
var China_Provinces = ee.FeatureCollection("users/wangjinzhulala/China_Provinces")
var China_List = China_Provinces.sort('Shape_Area',false).toList(10)
print(China_List)

var Area_No_1 = ee.Feature(China_List.get(0))
var Area_No_3 = ee.Feature(China_List.get(2))
var Area_No_7 = ee.Feature(China_List.get(6))

Map.centerObject(China_Provinces,4)
Map.addLayer(Area_No_1)
Map.addLayer(Area_No_3)
Map.addLayer(Area_No_7)
```

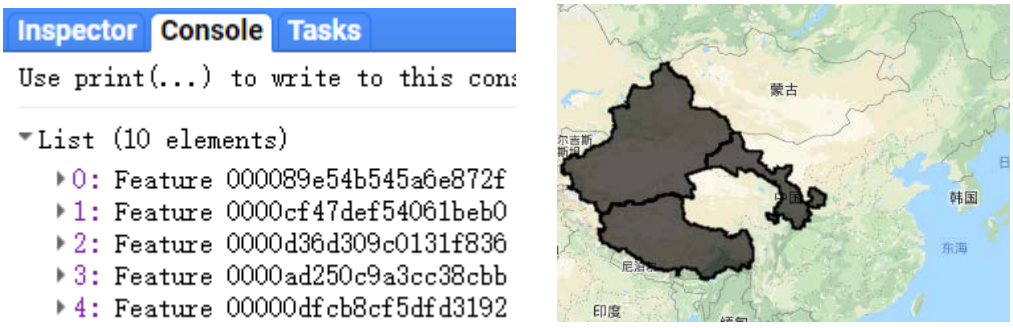


图 6.34 Feature Collection 的转 List 命令

前述及介绍过 List 是一个“数据框”。当 Feature Collection 转为 List 之后，我们就可通过.get()命令获取指定位置的 Feature。本例对 Feature Collection 进行排序并转 List 之后，获取了中国面积大小排第 1、第 3 和第 7 的三个省。

下边介绍 Feature Collection 属性统计命令，代码及执行效果如下：

```
var China_Provinces = ee.FeatureCollection("users/wangjinzhulala/China_Provinces")
var Area_of_No_1  = China_Provinces.sort('Shape_Area',false).aggregate_first('Shape_Area')
var Area_of_Top_3 = China_Provinces.sort('Shape_Area',false).limit(3)
                                   .aggregate_array('Shape_Area')
var Area_Status = China_Provinces.aggregate_stats('Shape_Area')
var Area_Histogram = China_Provinces.aggregate_histogram('Shape_Area')
print(Area_of_No_1)
print(Area_of_Top_3)
print(Area_Status)
print(Area_Histogram)
```

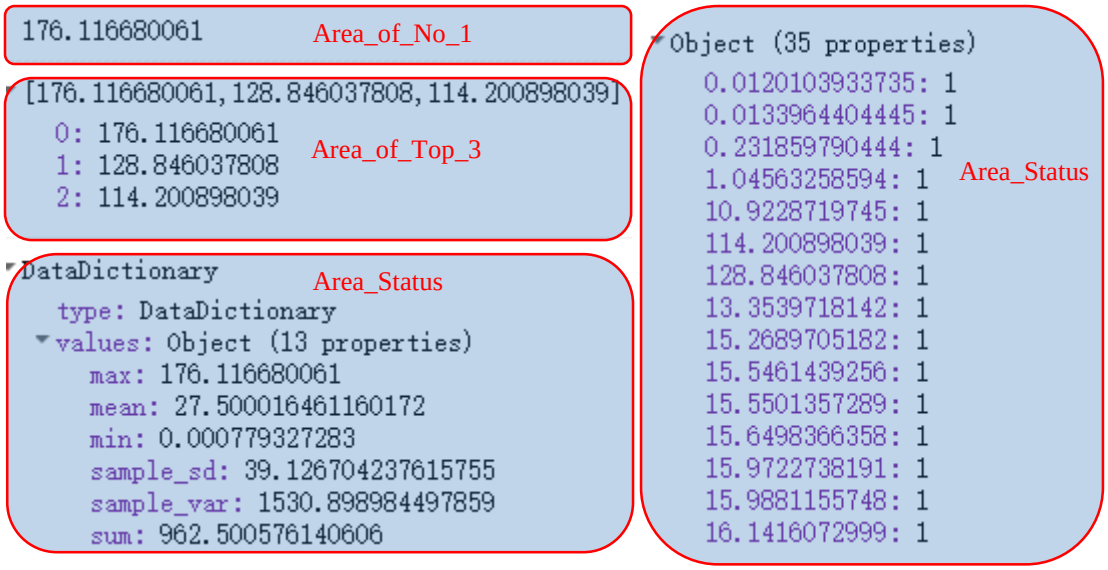


图 6.35 Feature Collection 的属性统计

下边是 GEE 中 Feature Collection 相关的属性统计命令：

图 6.36 Feature Collection 的全部属性统计命令

下边介绍 Feature Collection 的.map()命令，代码及执行效果如下：

```
var China_Provinces = ee.FeatureCollection("users/wangjinzhulala/China_Provinces")

function Add_Center (feature){
   return feature.centroid()
}

var China_Center = China_Provinces.map(Add_Center)
print(China_Center)

Map.centerObject(China_Provinces,4)
Map.addLayer(China_Provinces)
Map.addLayer(China_Center,{color:'ff0000'})
```

图 6.37 Feature Collection 的 map 命令

Feature Collection 的.map()命令可以看做是一种“集合操作”，本例的目的是对所有的 Feature 添加中心坐标，此时可以通过 function()命令编写一个计算中心点的操作，然后利用.map()命令将这个操作应用于整体 Feature Collection 中。值得一提的是，GEE 在处理数据时是基于“分布式计算”原理的，因此，当对某数据集中的元素进行“集合操作”时，GEE 能够充分利用计算资源，提高计算效率。

下边是本节介绍过的全部常用 Feature Collection 命令，尝试回忆其语法和功能：

ee.FeatureCollection()	.randomPoints()	.filterMetadata()	.limit()	.filterDate()
.filterBounds()	.filter()	.select()	.distinct()	.union()
.merge()	.set().remap()	.sort()	.makeArray()	.geometry()
.reduceToImage()	.first()	.toList()	.aggregate_first()	.aggregate_array()
.aggregate_stats/_histogram/_count/_count_distinct()				
.aggregate_max/_min/_sum/_mean/_product()				
.aggregate_sample_var/_total_var/_sample_sd/_total_sd/.map()				

本材料由王金柱（西南大学&迪肯大学）创作。如有需要请与我联系。
This doc contributed by Jinzhu Wang of Southwest University & Deakin University.
Email: wangjinzhulala@gmail.com