閸掆晝鏁ゆ妯虹棄閸︽澘娴楢PI閻㈢喐鍨氱悰灞炬杺閸栧搫鍨濋崶

閸涖劌宸

2018楠<9e>3閺<88><88>15閺<83>

### 介绍

在利用地图开展各种分析或制图时，准确的行政区划底图往往是不可或缺的。但由于我国目前行政区划（特别是区县级）调整比较频繁，互联网上的免费资源通常比较陈旧，只用这样的底图往往就不能准确反映现实情况了。好在高德地图提供了行政区域查询的WEB服务API[接口](http://lbs.amap.com/api/webservice/guide/api/district/)，可以通过这个接口查询我国各级行政区的行政区划地理空间信息，并利用返回的点坐标生成各级行政区划图。这样就可以利用高德地图的数据生产比较新的行政区划图了，而且高德说是“**唯一能让用户查询到乡镇/街道级别信息且小时级更新数据的公开API**”。惊喜不惊喜，意外不意外？废话不多说，下面我们就用R语言来即时生成行政区划底图吧。

### 抓取信息

library('httr')  
library('jsonlite')  
library('tidyverse')  
library('rlist')  
library('Rgctc2',lib.loc='~/GitHub/R\_coordination\_transformation')  
library('sf')

首先自然是写抓取信息的函数，利用高德的官方指南很容易搞定。

options(digits=11)  
get\_location<- function(address){  
 key = '7c6b6c0d1b641f4aa9cdb7d2229ae728' #需要预先申请一个高德API的key  
 url = 'http://restapi.amap.com/v3/config/district?' %>%  
 paste('keywords=' , address ,  
 '&key=' ,key ,  
 '&subdistrict=1' , #可以指定返回行政区的层级  
 '&extensions=all',  
 sep = '')  
 geoinfo<-GET(url)%>% content(as="text",encoding="UTF-8") %>%   
 fromJSON(flatten = TRUE) #返回信息的类型是可以选择的   
 return(geoinfo)  
}

这个爬虫就只需要一个参数*address*，支持中文地名，也支持数字形式的区域编码（adcode）。用中文地名查询时到了市、县这个层级就有可能碰到多义性的问题了，也就是一个关键字对应多个区域的情况，高德建议大家尽量使用adcode，可以在[这里](http://lbs.amap.com/api/webservice/download)下载。我们可以通过上级行政区域获得所有下级行政区域的adcode，非常简单，可以用这个机制直接抓取指定行政区里所有下级行政区的adcode。值得一提的是预定设定小数点位数options(digits=11)。由于地理坐标小数点后的位数比较多，如果不预先指定小数点位数的话，按照R默认的精度就会出现显示不完整的情况。下面，我们赶紧来拿浙江做个例子吧。

zj<-get\_location('浙江省') %>% '[['('districts')

返回的信息是一个包含了六个元素的列表，这个列表的*districts*元素包含了我所需要的所有信息，所以这里就直接一步提取出来了，有兴趣的小伙伴可以自己查看返回列表的信息。下面我们就来看看这个*districts*元素长什么样子吧。

## 'data.frame': 1 obs. of 7 variables:  
## $ citycode :List of 1  
## ..$ : list()  
## $ adcode : chr "330000"  
## $ name : chr "浙江省"  
## $ polyline : chr "121.134149,27.786010;121.134130,27.785898;121.134110,27.785840;121.134079,27.785817;121.134009,27.785782;121.13"| \_\_truncated\_\_  
## $ center : chr "120.152585,30.266597"  
## $ level : chr "province"  
## $ districts:List of 1  
## ..$ :'data.frame': 11 obs. of 6 variables:  
## .. ..$ citycode : chr "0572" "0571" "0573" "0579" ...  
## .. ..$ adcode : chr "330500" "330100" "330400" "330700" ...  
## .. ..$ name : chr "湖州市" "杭州市" "嘉兴市" "金华市" ...  
## .. ..$ center : chr "120.086809,30.89441" "120.209789,30.24692" "120.75547,30.746191" "119.647229,29.079208" ...  
## .. ..$ level : chr "city" "city" "city" "city" ...  
## .. ..$ districts:List of 11  
## .. .. ..$ : list()  
## .. .. ..$ : list()  
## .. .. ..$ : list()  
## .. .. ..$ : list()  
## .. .. ..$ : list()  
## .. .. ..$ : list()  
## .. .. ..$ : list()  
## .. .. ..$ : list()  
## .. .. ..$ : list()  
## .. .. ..$ : list()  
## .. .. ..$ : list()

为了比较清晰的展示数据结构，我选择只返回了下一级行政区的信息（subdistrict=1），也就是浙江的地级市信息。在实际中我设定subdistrict=3，可以一步返回浙江所有镇的信息。 *polyline*这个元素里就是我们要找的行政区边界上的坐标点的信息，目前是一个很长的字符串，后面主要的工作其实就是分割——转换类型——转换坐标系——生成地理空间对象，然后就可以作图啦。下面，我们就来处理浙江省的边界。

### 清洗、转换

## chr "121.134149,27.786010;121.134130,27.785898;121.134110,27.785840;121.134079,27.785817;121.134009,27.785782;121.13"| \_\_truncated\_\_

老规矩，先来看看*polyline*的结构。类型，字符串型（chr)；坐标是以经纬度的形式记录的，经纬度之间用**“，”**分隔，而每对坐标之间用**“；”**分隔。在后面，我们还看到了一个分隔符号**“|”**，这是什么意思呢？这表示着行政区并非只包括一个多边形。由于岛屿、飞地等类型的行政区，一个省的行政区可能会包括很多的独立多边形。每一个多边形之间都是用**“|”**来分隔。了解清楚了数据结构，就可以开始进行分割和转换了。下面给出代码。

zj$polyline<-zj$polyline %>%   
 str\_split('\\|') %>%   
 lapply(str\_split,';') %>%   
 '[['(1)%>% lapply(str\_split,',') %>% #以上对字符串进行分隔，得到一个多层嵌套的列表   
 lapply(lapply,as.numeric) %>%   
 lapply(list.rbind)%>% #转换数据类型，并合并行，形成若干矩阵。  
   
 lapply(gcj02\_wgs84\_matrix\_matrix) %>% #转换投影坐标系  
   
 lapply(list) %>%   
 st\_multipolygon %>% st\_sfc(crs=4326) #将polyline列定义为sfc属性  
zj\_sf<-st\_sf(zj) #将整个zj数据框定义为sf对象

这里要跳到开头讲一下三个包：

* 列表操作**rlist**包。这是R语言里对list对象进行操作的神器。由于list对象的非结构性，并且可以多层嵌套，所以一个字符串经过多次分割后就形成了有三四个层次深度嵌套的list对象，后面的工作无论是数据类型转换还是坐标系的转换都涉及到对list对象的深层操作。我基本上用的是**base**包里的lapply函数嵌套以应对，但是也少不了要借助**rlis t**包里很多函数。这段最后的代码里真正属于**rlist**包的函数实际上只有list.rbind一个了，但是实际上在调试的过程中我借助了很多这个包里的函数进行debug。如果对抓取网络地理信息数据有兴趣的话，应该都是要对抓下来的数据首先进行一番这样的操作的。
* 地理坐标转换**Rgctc2**包。出于保密的需要，高德、百度提供的坐标信息都是经过转换加密的，一般转换成wgs84投影坐标系通用性会更好。这种转换网上有各种语言的源代码也有各种小工具，但是目前R语言里好像还没有专门用于这个转换的包。于是我就自己简单写了一个包，可以在[这里](https://github.com/zhouqiangnju/R_coordination_transformation)下载。这个包功能很简单，目前就只能从高德转换到WGS84（完整的功能应该能实现高德、百度、WGS84中任意两种坐标系互转，而且在R中要用的方便话还要考虑各种数据类型的输入输出，所以还是个蛮大的工程，如果后面有兴趣就慢慢完善吧。因为不会，所以还没有搞帮助文档。）gcj02就是高德采用的坐标系，wgs84是我们要转换成的通用坐标系,matrix\_matrix分别是输入和输出的数据类型。这个包里所有的函数基本上都是这个命名规则。
* 生成地理空间对象**sf**包。它是**sp**包的升级换代，相比于传统的**sp**包属性层和地理空间信息层分隔的复杂存储方法，**sf**包就是基于数据框的，非常便于在R语言中进行操作，也是更加主流的GIS数据存储方式。由于目前得到了ggplot2的支持，专门加了一个对象geom\_sf用于**sf**对象的作图，所以目前看来前途非常光明啊。 这里可以再看一下zj这个数据框，实际上只要把*polyline*这一列转换成**sf**对象必需的*sfc*列（也就是专门用于存储地理空间信息的列），整个zj数据框就可以定位为**sf**对象了，非常方便。st\_sfc(crs=4326)这个命令就是用来转换*polyline*列的。转换完成后我们再看*polyline*列的属性

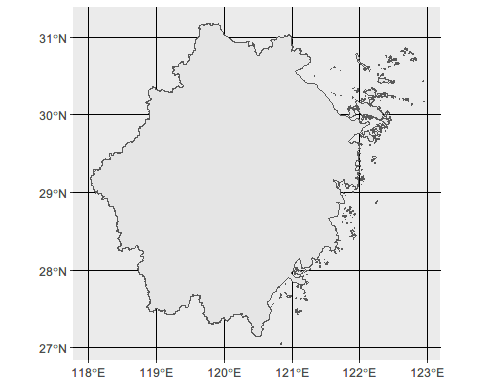
## $class  
## [1] "sfc\_MULTIPOLYGON" "sfc"   
##   
## $precision  
## [1] 0  
##   
## $bbox  
## xmin ymin xmax ymax   
## 118.023019231 27.044744711 122.945311212 31.180867236   
##   
## $crs  
## Coordinate Reference System:  
## EPSG: 4326   
## proj4string: "+proj=longlat +datum=WGS84 +no\_defs"  
##   
## $n\_empty  
## [1] 0

可以看到，已经附加了很多地理空间信息。 不方便的地方在于，不同的地理空间对象类型（就是点、线面）所对应的*sfc*列对数据格式的要求不同而且比较严格，比如说我这里所采用的*multipolygon*这个对象，他只接受*list(list(matrix),list(matirx),……)*这种格式的数据,而这里的*matrix*是由坐标对构成的矩阵，所以也就是*matrix(c(lng1,lat2),c(lng2,lat3)…..)*，是一个一维的矩阵，在R里面生成这样一个矩阵还是有点技巧的，需要一定时间来熟悉这个包的特殊用法。

### 成图

最艰难的阶段已经过去啦，下面就是愉快的画图啦。加载**ggplot2**，一行命令就出图。

library(ggplot2)  
ggplot()+geom\_sf(data=zj\_sf)

 ###延伸 通过以上的方法，我们可以一个行政区一个行政区的得到他们的边框，然后通过rbind得到更大区域的图了。然而，这样的方式是太low了。我们要充分利用R语言里的apply函数家族，一下子得到浙江省所有地级市的行政区图。直接上代码。

zj\_city<-zj %>% "["('districts') %>% '[['(1) %>% '[['(1) #提取各城市adcode  
zj\_city<-lapply(zj\_city$adcode,get\_location) %>% #利用lapply函数提取所有城市的地理信息  
 list.map(districts) %>%   
 lapply(select,-districts) %>%   
 list.rbind  
zj\_city$polyline <- zj\_city$polyline %>%   
 str\_split('\\|') %>%   
 lapply(str\_split,';') %>%   
 lapply(lapply,str\_split,',') %>%  
 lapply(lapply,lapply,as.numeric) %>%   
 lapply(lapply,list.rbind) %>%  
 lapply(lapply,gcj02\_wgs84\_matrix\_matrix) %>%   
 lapply(lapply,list) %>%  
 lapply(st\_multipolygon) %>%st\_sfc(crs=4326)  
  
zj\_city <- st\_sf(zj\_city)

可以看出来，最后一段代码不过是又套了一层lapply而已。在这里我们先不着急出图，我们把各城市的名字标出来。位置就放在高德给出的城市中心位置，原汁原味吗。这里又要进行一下坐标的转换。

zj\_city <-zj\_city$center %>%   
 str\_split(';') %>%   
 lapply(str\_split,',') %>%   
 lapply(lapply,as.numeric) %>%   
 lapply(list.rbind) %>% list.rbind %>%   
 gcj02\_wgs84\_matrix\_df %>%  
 bind\_cols(zj\_city)  
zj\_city<-st\_sf(zj\_city)

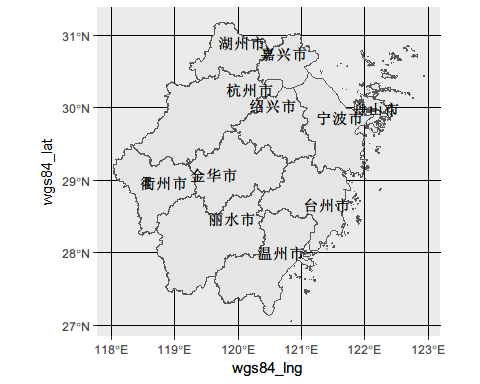
我们来检验一下最终成果吧。

library(showtext)

## Loading required package: sysfonts

## Loading required package: showtextdb

showtext\_auto() #由于ggplot2对中文支持不友好，这里需要加载一个显示中文的包  
ggplot() + geom\_sf(data=zj\_city) +geom\_text(data=zj\_city,aes(x=wgs84\_lng,y=wgs84\_lat,label=name))



浙江省分市行政区划图