Subject: New tool: sort points by countries with a shapefile and IDL
Posted by MarioIncandenza on Thu, 17 Jun 2010 19:59:09 GMT
View Forum Message <> Reply to Message

This is something I've wanted to be able to do in IDL for quite some time. The obstacles were that the actual country polygon data were only available through ESRI, under a "permissive but definitely not free" license, and that shapefile usage in IDL kinda scared me. I overcame the second myself, but the really big deal is new free GIS layers from Natural Earth: http://www.naturalearthdata.com/ , filling a huge gap in open-source GIS.

The routines below are designed to take XY input and sort by country. They require the "scale ranks" polygon layer from Natural Earth:
http://www.naturalearthdata.com/downloads/110m-cultural-vect ors/110m-admin-0-details/

I won't claim will do everything you might want, but with these tools you can easily roll your own to do whatever you need with these data. There are two functions here, one to ingest the polygon shapefile, and another to match it to the XY data. These routines require UNDEFINE.PRO from the Coyote library, but that should be the only dependency (I think).

This version uses the 110M:1 coarse-resolution data, and can process a million XY points in about an hour. You can also use this routine to select points within a single country, which is much faster. Suggestions for speeding this up are welcome.

```
function
 ingest_naturalearth_admin0_110m_scaleranks_shapefile,shapefi le_path
 ;############################################################## #############
; Extract the country polygon dataset
 ;############################################################## #############
;NAME: ingest_naturalearth_admin0_110m_scaleranks_shapefile()
;NOTES: The Natural Earth project is a volunteer-driven, public-domain
;project to produce usable GIS layers for basic cultural and physical
;parameters. The data used by this routine is the ADMIN0 political
;map, in the 'scale rank' configuration where each polygon has clean
;topology (==no intersections), and so can be used as an ROI.
;To download the shapefile, and find more information, go to
; http://www.naturalearthdata.com/
;  http://www.naturalearthdata.com/downloads/110m-cultural-vect ors/110m-admin-0-details/
;PURPOSE: extract country 'scale rank' polygons
;OPTIONAL KEYWORD: SHAPEFILE_PATH = location of shapefile on disk.
;RETURN VALUE: Structure with attributes and polygon vertices
;USAGE:
 POLY_STRUCT=INGEST_NATURALEARTH_ADMIN0_10M_SCALERANKS_SHAPEF
ILE([SHAPEFILE_PATH])
```

```
;DEPENDENCIES: UNDEFINE.PRO (Coyote)
country_shapes=create_struct(name='COUNTRY_POLY_110M', $
                        "ScaleRank",0l, $ ; X
                        "FeatureCla","", $ ; X
                        "SOVEREIGNT","", $ ; SOV in output from
SORT_COUNTRIES
                        "SOVISO","", $
                        "SOV_A3","", $ ; SOV3 in output
                        "LEVEL",0.0d, $
                        "TYPE","", $
                        "NAME","", $
                        "SORTNAME","", $
                        "ADM0_A3","", $
                        "GEOUNIT","", $ ; called MAPUNIT in
output
                        "GU_A3","", $
                        "SUBUNIT","", $ ; X
                        "SU_A3","", $
                        "NAME_SM","", $
                        "NAME_LNG","", $
                        "TERR_","", $
                        "PARENTHETI","", $
                        "NAME_ALT","", $
                        "LOCAL_LNG","", $
                        "LOCAL_SM","", $
                        "FORMER","", $
                        "ABBREV_","", $
                        "MAP_COLOR",0.0d, $
                        "PEOPLE",0.0d, $
                        "GDP_USDM",0.0d, $
                        "FIPS_10","", $
                        "ISO_A2","", $
                        "ISO_A3","", $
                        "ISO_N3",0.0d, $
                        "ITU","", $
                        "IOC","", $
                        "FIFA","", $
                        "DS","", $
                        "WMO","", $
                        "GAUL",0.0d, $
                        "MARC","", $
                        "STANAG1059","", $
                        "GW_ID",0.0d, $
                        "DIAL",0.0d, $
                        "INTERNET_","", $
                        "COG","", $
                        "ACTUAL","", $
                        "CAPAY","", $
```

```
                        "CRPAY","", $
                        "ANI","", $
                        "LIBENR","", $
                        "ANCNOM","", $
                        "PAYS_R_GIO","", $
                        "COMMENT","", $
                        "polyx",ptr_new(), $
                        "polyy",ptr_new())
if n_elements(shapefile_path) eq 0 then $
  country_shapefile='110m_admin_0_scale_ranks.shp'
else country_shapefile=shapefile_path
if(file_test(country_shapefile,/read) eq 0) then
message,string(country_shapefile,format='("Can''t read ",a)')
myshape=OBJ_NEW('IDLffShape',country_shapefile)
if(size(myshape,/type) ne 11) then message,"Couldn''t create
IDLffShape object!"
myshape->IDLffShape::GetProperty, N_ENTITIES=n_ent
myshape->IDLffShape::GetProperty, ATTRIBUTE_NAMES=attr_names ; not
strictlyt necessary, included for reference
nattr=n_elements(attr_names)
out=replicate(country_shapes,n_ent)
for i_ent = 0l,n_ent-1 do begin
;out=replicate(country_shapes,3)
;for i_ent = 0l,2 do begin
    attr=myshape->IDLffShape::GetAttributes(i_ent)
    for i_attr=0l,nattr-1 do out[i_ent].(i_attr) = attr.(i_attr)
    ent=myshape->IDLffShape::GetEntity(i_ent)
    if(ent.shape_type ne 5) then begin
        print,"non-polygon! Boom!"
        print,attr.attribute_3
        continue
    endif
    out[i_ent].polyx=ptr_new(reform((*ent.vertices)[0,*]))
    out[i_ent].polyy=ptr_new(reform((*ent.vertices)[1,*]))
    undefine,ent
endfor
obj_destroy,myshape
return,out
end

function
 sort_countries_110m,xdata,ydata,shapefile_path=shapefile_pat
h,country_list=country_list,poly_struct=poly_struct,target=t arget
compile_opt idl2
on_error,0
 ;############################################################# #############
; Do some operations with the country polygon dataset
 ;############################################################# #############
```

```
;NAME: sort_countries()
;PURPOSE: extract country 'scale rank' polygons and evaluate them
;against XY input.
; USAGES:
;   1) No XY DATA: returns structure giving information on geographic
;   units: COUNTRY_LIST =
SORT_COUNTRIES([SHAPEFILE_PATH=SHAPEFILE_PATH])
;   2) XY DATA, no TARGET: Returns structure giving geographic fields
;   for each XY point (SLOW and INTENSIVE)
;   XY_GEO_STRUCT=SORT_COUNTRIES(XDATA,YDATA,
[SHAPEFILE_PATH=SHAPEFILE_PATH])
;   3) XYDATA + TARGET: finds polygons with TARGET[0] = TARGET[1],
;   returns 0|1|2|3 = outside/inside/edge/vertex (see documentation
;   for IDLanROI for details):
;   InCountryB = SORT_COUNTRIES(XDATA,YDATA,TARGET=[FIELD,VALUE])
;ARGUMENTS:
;   XDATA,YDATA: Longitude/Latitude points for comparison against
;   shapefile
;OPTIONAL KEYWORDS:
;   SHAPEFILE_PATH: Location of the country polygon shapefile on disk.
;   COUNTRY_LIST: structure of geographic fields (see Usage 1)
;   POLY_STRUCT: complete structure of shapefile data.
;   TARGET = ['Field','Value']. $FIELD is the attribute in the
;   shapefile that will be queried for $VALUE.
;   E.g. TARGET=['SOV','Vietnam']
;RETURN VALUE:
;   Variable. See USAGES.
;FILESYSTEM INPUT: Reads shapefile from $SHAPEFILE_PATH or default
;FILESYSTEM OUTPUT: NONE

;1) read in shapefile data
 poly_struct=ingest_naturalearth_admin0_110m_scaleranks_shape file(shapefile_path)
;2) create COUNTRY_LIST structure
i_uniq=uniq(poly_struct.subunit,sort(poly_struct.subunit))
nuniq=n_elements(i_uniq)
country_list1=create_struct(name='COUNTRY_LIST', $
                        "objectid",0l, $
                        "scalerank",0l, $
                        "featureclass","", $
                        "sov","", $
                        "sov3","", $
                        "admin_0","", $
                        "mapunit","", $
                        "subunit","")
i_attr_keep = [0,0,1,2,4,9,10,12]; map from 110m scale-rank attributes
to above list
country_list=replicate(country_list1,nuniq)
for i_poly=0l,nuniq-1 do for i_attr=0,6 do country_list[i_poly].
```

```
(i_attr)=poly_struct[i_uniq[i_poly]].(i_attr_keep[i_attr]) ; paste
attributes
if n_elements(xdata) eq 0 then begin
   if(~arg_present(poly_struct)) then undefine,poly_struct
   if(arg_present(poly_struct)) then print,"giving back POLY_STRUCT"
   return,country_list
endif
if(n_elements(xdata) ne n_elements(ydata)) then message,"XDATA and
YDATA don"t match!"
;3) Check for TARGET
if n_elements(target) ne 0 then begin
   if n_elements(target) ne 2 then message,"Usage: TARGET =
[FIELD,VALUE]"
   target_tag=target[0] & target_name=target[1]
   all_tags=tag_names(poly_struct)
   f_target_tag = where(strmatch(all_tags,target_tag,/
fold_case),nf_target_tag)
   if(nf_target_tag eq 0) then
message,string(target_tag,all_tags,format='("No ",a," in ",150(a))')
   if(nf_target_tag gt 1) then message,"WTF multiple tag matches!"
; OK, we have TARGET_TAG, now check for TARGET_NAME
   f_target = where(strmatch(poly_struct.(f_target_tag),target_name,/
fold_case),nf_target)
   if(nf_target eq 0) then
message,string(target_name,target_tag,format='("No ",a," in ",a)')
; OK, we have F_TARGET. Now query the polygons
   intarget=xdata * 0b
   for i_poly=0l,nf_target-1 do begin

 roi=obj_new('IDLanROI',*(poly_struct[f_target[i_poly]].polyx ),*(poly_struct[f_target[i_poly]].polyy))
      result=roi->IDLanROI::ContainsPoints(xdata,ydata)
      obj_destroy,roi
      intarget = intarget > result ; take highest value
   endfor
   if(~arg_present(poly_struct)) then undefine,poly_struct
   return,intarget
endif

;4) No TARGET specified. Do the complete scan. This can be VERY time-
consuming.
npoly=n_elements(poly_struct)
ndata=n_elements(xdata)
country_match1=create_struct(name='COUNTRY_MATCH', $
                        "objectid",0l, $
                        "scalerank",0l, $
                        "featureclass","", $
                        "sov","", $
                        "admin_0","", $
```

```
                        "mapunit","", $
                        "subunit","", $
                        "n_poly",0l) ; extra tag for multiple
matches
;NOTE: the fields will be the LAST MATCH. The N_POLY field is
;diagnostic of multiple matches.
country_match_all=replicate(country_match1,ndata) ; BIG
for i_poly=0l,npoly-1 do begin
;Method1: IDLanROI (loop)
   print,poly_struct[i_poly].subunit ; testing

 roi=obj_new('IDLanROI',*(poly_struct[i_poly].polyx),*(poly_s truct[i_poly].polyy))
   result=roi->IDLanROI::ContainsPoints(xdata,ydata)
   obj_destroy,roi
   country_match_all.n_poly = country_match_all.n_poly + (result eq
1)
   f_inside=where(result eq 1,nf_inside) ; INSIDE only: no edge/
vertex pts.
   if(nf_inside gt 0) then $
     for i_attr=0,6 do country_match_all[f_inside].
(i_attr)=poly_struct[i_poly].(i_attr_keep[i_attr]) ; paste attributes
;M2;; Method 2: INSIDE() (array, but chunked)
;M2;    veclen=n_elements(*(poly_struct[i_poly].polyx))
;M2;    maxarray=15e6               ;
;M2;    chunk= floor(maxarray / veclen) ; max array size
;M2;    n1k=(ndata/chunk)+1
;M2;    print,poly_struct[i_poly].subunit," Chunk = ",chunk, "N1k =
",n1k ; testing
;M2;    for i1k=0l,n1k-1 do begin
;M2;        print,i1k; testing
;M2;        f1k0=(i1k*chunk)
;M2;        f1k1=((i1k*chunk)+chunk-1) < (ndata-1)
;M2;        x1k=xdata[f1k0:f1k1]
;M2;        y1k=ydata[f1k0:f1k1]
;M2;
 result=inside(x1k,y1k,*(poly_struct[i_poly].polyx),*(poly_st ruct[i_poly].polyy))
;M2;        country_match_all[f1k0:f1k1].n_poly =
country_match_all[f1k0:f1k1].n_poly + (result eq 1)
;M2;        f_inside=where(result eq 1,nf_inside) ; INSIDE only: no
edge/vertex pts.
;M2;        if(nf_inside gt 0) then begin
;M2;            f_in1k=(indgen(f1k1-f1k0+1)+f1k0)[f_inside]
;M2;            for i_attr=0,6 do country_match_all[f_in1k].
(i_attr)=poly_struct[i_poly].(i_attr) ; paste attributes
;M2;        endif
;M2;    endfor                    ; end METHOD 2
endfor
if(~arg_present(poly_struct)) then undefine,poly_struct
```

```
return,country_match_all
end
```