

CSI606

Maps in PV-WAVE

PV-WAVE Mapping Procedures

- **map** - plots map data with a specified projection
- **map_contour** - plots contours on a map
- **map_plots** - plots vectors or points on the current map projection
- **map_polyfill** - fills specified regions of a map
- **map_reverse** - converts x-y coordinate data to longitude and latitude data
- **map_velovect** plots a two-dimensional vector field on a map
- **map_xyouts** - adds text to a map
- **usgs_names** - queries a database of longitude/latitude coordinates for states, counties, cities, and towns in the United States

PV-WAVE and MAP DATA SETS - I

- **A map in PV-WAVE is merely a set of polylines**
- **World Databank II Data set**
 - **Subset of a public domain U.S. Department of Commerce data set**
 - **Attributes**
 - **Group**
 - **coastlines, islands, lakes, rivers, international boundaries, and U.S. state boundaries**
 - **Areas**
 - **Asia, Europe, North America, and South America**
- **USGS Digital Line Graph Data set**
 - **Polygons for U.S. states and counties**

More PV-WAVE Map Data Set Information

- **USGS name Database**
 - **Lets one find the latitude and longitude for most populated cities and towns in the U. S.**
- **It is possible to read in your own customized maps into PV-WAVE**

A Simple World Map

```
WAVE> tek_color
```

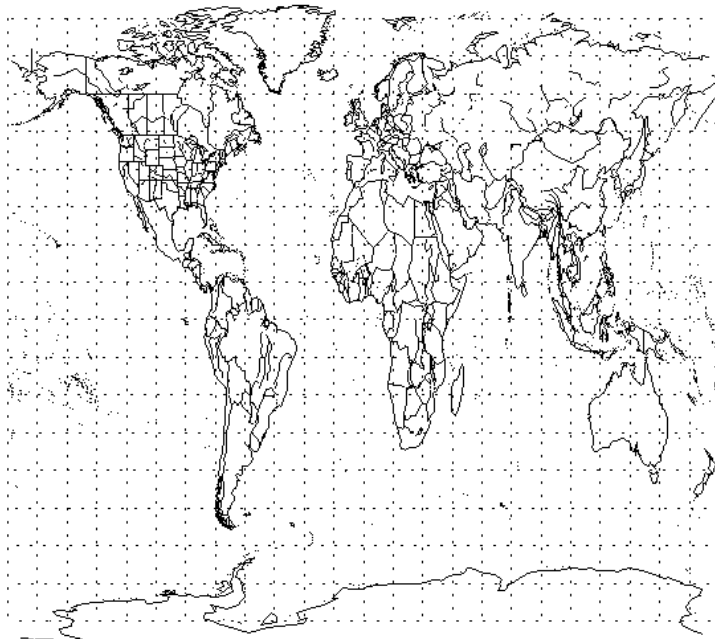
```
;range is min long, min lat, max long, max lat
```

```
WAVE> map,range=[-180,-90,180,90],$
```

```
- select={,group:['cil','bdy','pby'],$
```

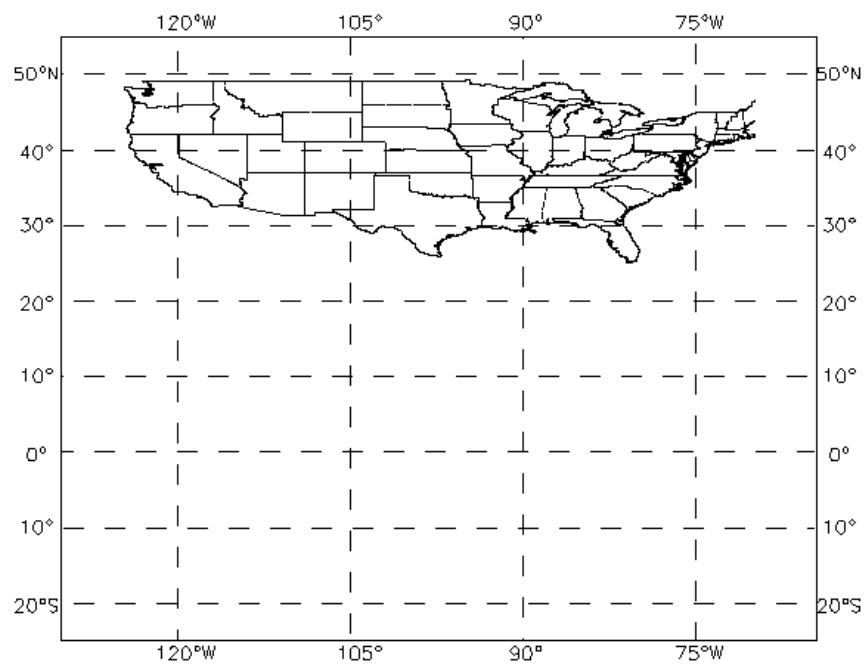
```
- 'riv'],area:''},color=16777215,/gridlines,$
```

```
- gridstyle=1 gridcolor=16777215
```



A US Map

```
WAVE> map,data='usgs_db',range=[-125,-25,-70,55],$  
-/gridlines,/axes,gridstyle=2,gridcolor=16777215,color=16777215
```



Map Projections

- **1 - Equidistant Cylindrical**
 - **2 - Lambert Conformal Conic**
 - **3 - Cylindrical Mercator**
 - **4 - Sinusoidal**
 - **5 - Albers Equal-Area Conic**
 - **6 - Polyconic**
 - **7 - Polar Stereographic**
 - **8 - Oblique Stereographic**
 - **9 - Oblique Orthographic**
 - **10 - Polar Orthographic**
 - **11 - Oblique Azimuthal Equidistant Oblique**
 - **12 - Polar Azimuthal Equidistant Oblique**
 - **13 - Polar Azimuthal Equal-Area**
 - **14 - Oblique Azimuthal Equal-Area**
 - **15 - Transverse Mercator**
 - **16 - Mollweide (Ellipsoid)**
 - **99 - Satellite (3D mapping onto a sphere)**
 - **-1 User-defined projection**
 - **0 - No Projection**
-
- **Map Specification**
 - **map, Projection = 4**

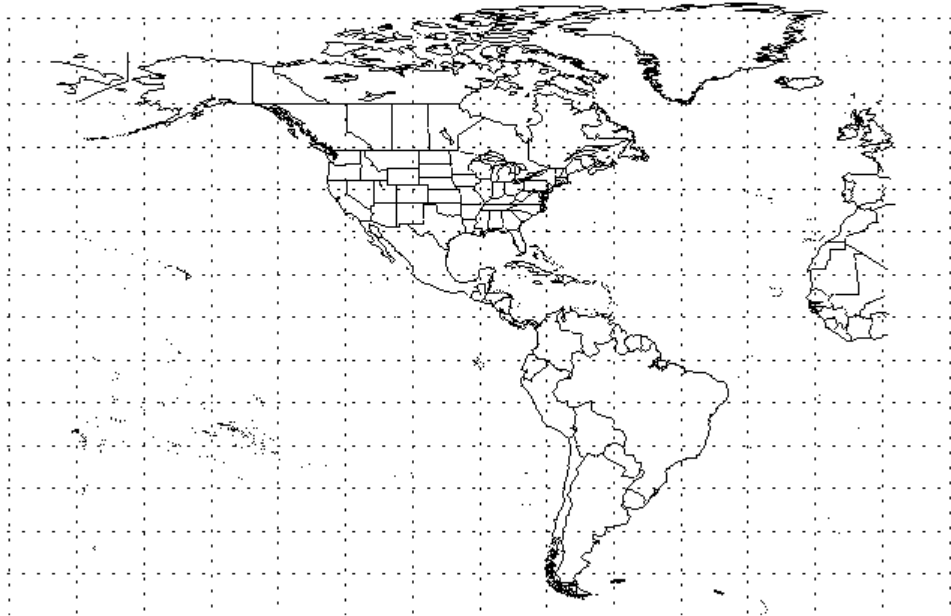
Selecting Map Attributes

- **WAVE> map,select = {, group: ['cil', 'riv'],area: 'europe'},color=16777215**



Using Center and Zoom

- **WAVE> map, center=[-90,30],zoom = 2,
/**



Plotting Great Circles and Computing Distances

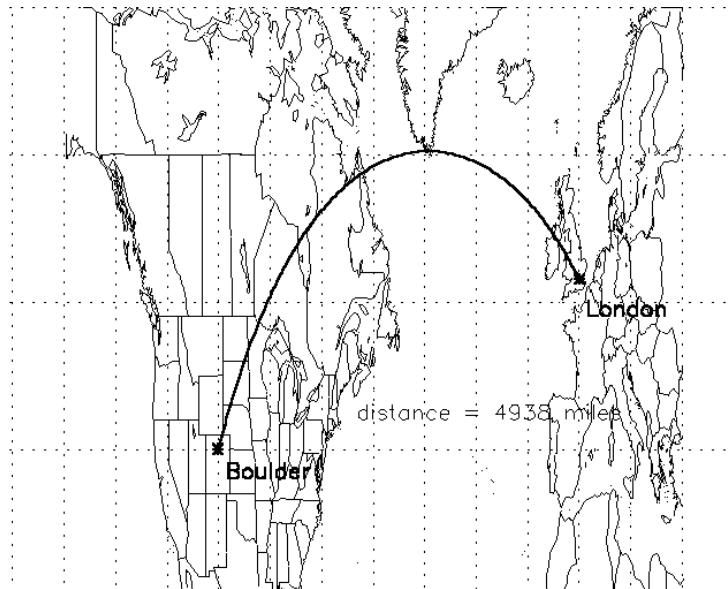
```
WAVE> map,range=[-150,30,30,70],/gridlines,gridcolor=16777215,gridstyle=1
```

```
WAVE> map_plots,[-105.3,-0.1],[40.0,51.5],$  
- distance=d,/miles,color=16777215,$  
- psym=-2,thick=2
```

```
WAVE> map_xyouts,-103.0,38.0,'Boulder',$  
- color=16777215,charsize=1.5,charthick=2
```

```
WAVE> map_xyouts,2.0,49.0,'London',$  
- color=16777215,charsize=1.5,charthick=2
```

```
WAVE> map_xyouts,-65.0,42.0,'distance =' + $  
- strcompress(string(d(0)),$  
- format='(I5)')+ ' miles',$  
- color=16777215,charsize=1.5
```



Adding an Image Under a Map

```
WAVE> file='c:\vni\mapping-1_1\data\'+'$  
- 'earth_elev.dat'  
WAVE> elev=FLTARR(720,360)  
WAVE> openr,1,file,/xdr  
WAVE> readu,1,elev  
WAVE> close,1  
WAVE> window,xsize=720,y size=360,colors=128  
WAVE> tvlct,150,150,150,63  
WAVE> red=bytarr(63)  
WAVE> grn=bytsc1(findgen(63)^2.0)  
WAVE> blu=bytsc1((findgen(63)))  
WAVE> tvlct,red,grn,blu,0  
WAVE> tvlct,255,255,255,127  
WAVE> map,projection=4,range=[-180,-90,180,90],  
- image=elev
```

