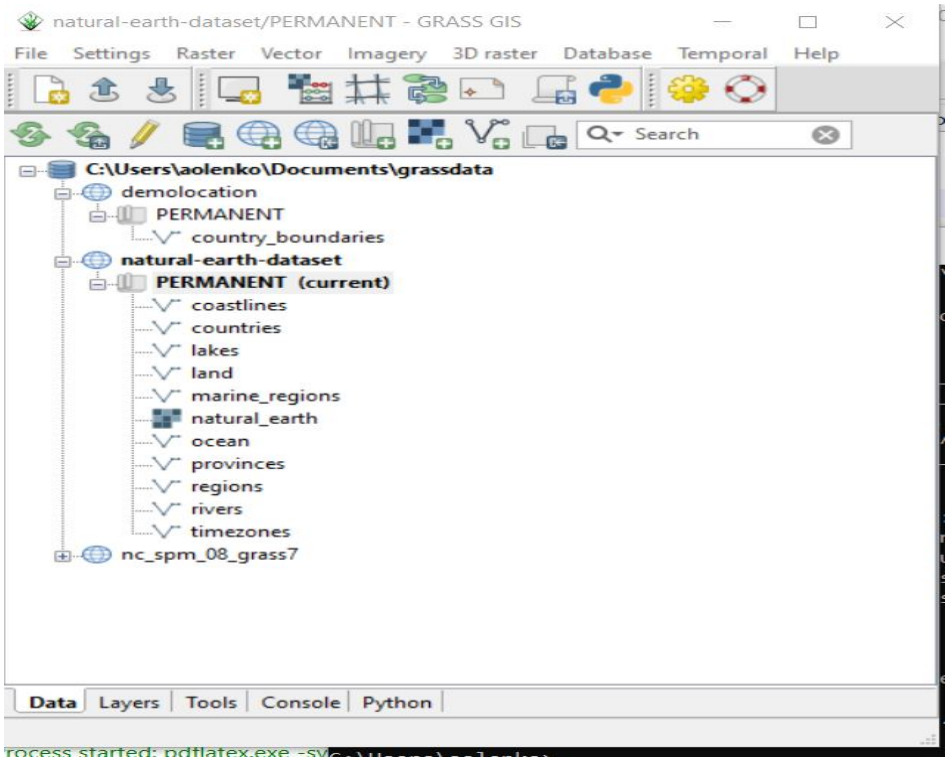
Week 4 (part 1): Integration of R with GRASS (using world map)

Example 3.

Let us consider another example based on week 3 materials. Now we will plot cities with largest population on the world map in GRASS.
Start GRASS with the LOCATION "natural-earth-dataset" and the MAPSET "Permanent".



Then, launch R within the GRASS terminal typing in Console :

Rgui

Then, load the R/GRASS interface as follows:

```
> library(rgrass7)
> library(rgdal)
> library(sp)
> use_sp()
```

We will use the dataset worldcities.csv from the World Cities Database(`https://simplemaps.com/data/world-cities`) with information about large cities across the world.
You can also download it from the LMS folder Data.

```
> cities <- read.csv("D:/worldcities.csv", header = TRUE)
> str(cities)
'data.frame':   12893 obs. of  11 variables:
$ city      : chr  "Malishev" "Prizren" "Zubin Potok" ...
$ city_ascii: chr  "Malisheve" "Prizren" "Zubin Potok"  ...
$ lat       : num  42.5 42.2 42.9 42.6 42.3 ...
$ lng       : num  20.7 20.7 20.7 21.6 21.4 ...
$ country   : chr  "Kosovo" "Kosovo" "Kosovo" "Kosovo" ...
$ iso2      : chr  "XK" "XK" "XK" "XK" ...
$ iso3      : chr  "XKS" "XKS" "XKS" "XKS" ...
$ admin_name: chr  "Malishev" "Prizren" "Zubin Potok"  ...
$ capital   : chr  "admin" "admin" "admin" "admin" ...
$ population: num  NA NA NA NA NA NA NA NA NA NA ...
$ id        : int  1901597212 1901360309 1901608808   ...
> cities <-cities[complete.cases(cities), ]
```

We consider only the cities with population more than 10000000:

```
> sum(cities$population > 10000000)
[1] 19
> mcities<-cities[cities$population > 10000000, ]
> mcities<-mcities[, c("lng","lat", "population")]
```

```
> mcities<-mcities[, c("lng","lat", "population")]
> mcities
             lng       lat population
1071    -99.1310  19.4424   19028000
1683    120.9822  14.6042   11100000
1786     66.9900  24.8700   12130000
2216     37.6155  55.7522   10452000
3390     29.0100  41.1050   10061000
4218    -58.3975 -34.6025   12795000
4530     90.4086  23.7231   12797394
5045    -43.2250 -22.9250   11748000
5097    -46.6250 -23.5587   18845000
5736    121.4365  31.2165   14987000
5960    116.3883  39.9289   11106000
6538     31.2500  30.0500   11893000
7441     72.8570  19.0170   18978000
7468     88.3247  22.4950   14787000
7512     77.2300  28.6700   15926000
7812    139.7514  35.6850   35676000
7849    135.4601  34.7500   11294000
9739    -73.9249  40.6943   19164071
10410 -118.4068  34.1140   12740381
```

We rescale the population in 10000000 and create SpatialPointsDataFrame using cities' locations and rescaled population:

```
> mcities$population <- mcities$population/10000000
> mcities_coor <- cbind(mcities$lng, mcities$lat)
> row.names(mcities_coor) <-  1:nrow(mcities_coor)
> row.names(mcities) <-  1:nrow(mcities)
> str(mcities_coor)
> llCRS <- CRS("+proj=longlat +ellps=WGS84")
> mcities_sp <- SpatialPoints(mcities_coor, proj4string = llCRS)
> summary(mcities_sp)
```

```
> mcities_sp <- SpatialPoints(mcities_coor, proj4string = llCRS)
> summary(mcities_sp)
Object of class SpatialPoints
Coordinates:
             min      max
coords.x1 -118.4068 139.7514
coords.x2  -34.6025  55.7522
Is projected: FALSE
proj4string : [+proj=longlat +ellps=WGS84 +no_defs]
Number of points: 19
```

Then we create a spatial data frame by using

```
> mcities_spdf <- SpatialPointsDataFrame(mcities_coor,
+ mcities,proj4string = llCRS, match.ID = TRUE)
> summary(mcities_spdf)
```

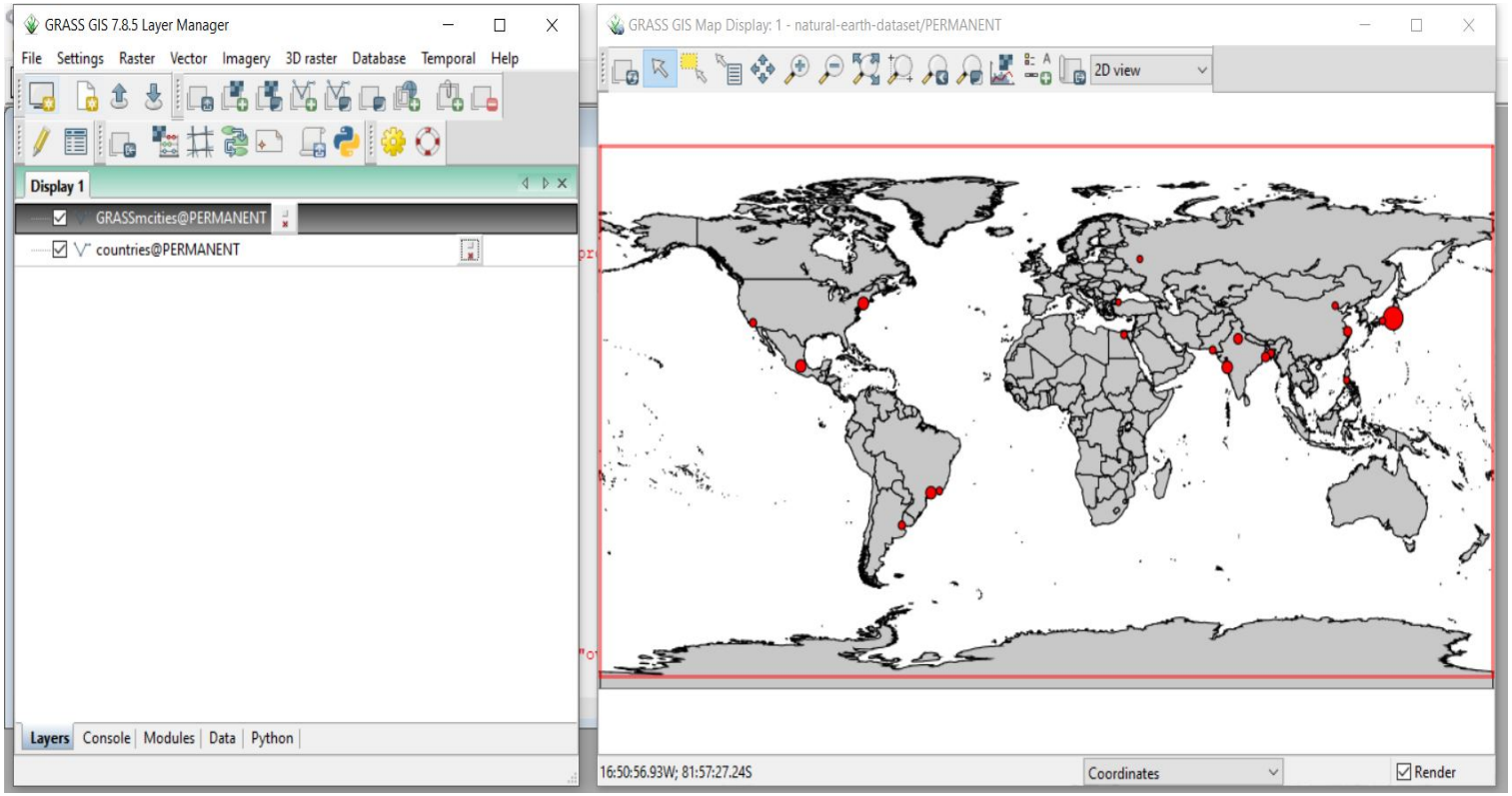
```
> mcities_spdf <- SpatialPointsDataFrame(mcities_coor, mcities,proj4string = ll$
> summary(mcities_spdf)
Object of class SpatialPointsDataFrame
Coordinates:
               min      max
coords.x1 -118.4068 139.7514
coords.x2  -34.6025  55.7522
Is projected: FALSE
proj4string : [+proj=longlat +ellps=WGS84 +no_defs]
Number of points: 19
Data attributes:
      lng                lat           population
 Min.   :-118.41   Min.   :-34.60   Min.   :1.006
 1st Qu.: -44.92   1st Qu.: 19.23   1st Qu.:1.152
 Median :  66.99   Median : 28.67   Median :1.280
 Mean   :  36.21   Mean   : 21.84   Mean   :1.503
 3rd Qu.: 103.40   3rd Qu.: 35.22   3rd Qu.:1.739
 Max.   : 139.75   Max.   : 55.75   Max.   :3.568
>
```

Finally we save the obtained SpatialPointsDataFrame into GRASS as the vector dataset GRASSmcities:

```
> writeVECT(mcities_spdf,"GRASSmcities",
+ v.in.ogr_flags=c("o", "overwrite"), ignore.stderr=TRUE)
```

Then we reload the GRASS location and plot GRASSmcities. In addition we plot the boundaries of countries, which are in the vector layer COUNTRIES.

To modify how the cities are shown double click on the layer GRASSmcities, change Colours to "red", Symbols to "circle" and in Symbols' the field "name of numeric column containing symbol size" to "population".


GRASS GIS 7.8.5 Layer Manager
File Settings Raster Vector Imagery 3D raster Database Temporal Help
Display 1
GRASSmcities@PERMANENT
countries@PERMANENT
Layers Console Modules Data Python
GRASS GIS Map Display: 1 - natural-earth-dataset/PERMANENT
2D view
16:50:56.93W; 81:57:27.24S
Coordinates
Render