Week 3: GRASS and R

This week gives an overview of GRASS and its interface with R. In particular, we consider

- **GRASS database and command structure**.
- GRASS **command line and GUI interface**.
- GRASS **visualisation tools**.
- Examples of **data import** in GRASS.
- **Data vectorisation** in GRASS.
- **Integration of R with GRASS.**
- Example of **North Caroline data.**
- Example of **importing CRAN data.**

GRASS.

GRASS (Geographical Resources Analysis Support System) is a raster/vector GIS combined with integrated image processing and data visualization subsystems. It includes approximately 400 modules for management, processing, analysis and visualization of georeferenced data.

GRASS, as a multipurpose GIS, with data organized as raster and vector maps, provides a wide range of tools to support most of the GIS functionality.

The full GRASS reference manual can be found at `https://grass.osgeo.org/learn/manuals/`

GRASS database and command structure

GRASS data are stored in a directory referred to as **GIS DataBASE.** This directory must be created before you start working with GRASS. You can create it either in your home directory or in a shared network directory.

Within this directory, the GRASS GIS data are organized by projects stored in subdirectories called **LOCATIONs.** Each LOCATION is defined by its coordinate system, map projection and geographical boundaries. The subdirectories and files defining a LOCATION are created automatically when GRASS is started for the first time with a new LOCATION.

Each LOCATION can have several **MAPSETs** (subdirectories of the LOCATION) that are used to subdivide the project into different topics, subregions, or as workspaces for individual team members. Besides access to his own MAPSET, each user can also read maps in other users MAPSETs, but they can modify or remove only the maps in their own MAPSET.

GRASS/GIS functionality

geospatial data integration	import and export of data in various formats coordinate systems transformations and projections transformations between raster and vector data 2D/3D spatial interpolation and approximation
2D/3D raster data processing	2D and 3D map algebra surface and volume geometry analysis topographic parameters and landforms flow routing and watershed analysis line of sight, insolation cost surfaces, shortest path, buffers landscape ecology measures correlation, covariant analysis expert system (Bayes logic)
2D/3D vector data processing	multi-attribute vector data management digitizing overlay, point and line buffers vector network analysis spatial autocorrelation summary statistics multivariate spatial interpolation and approximation Voronoi polygons, triangulation

image processing	processing and analysis of multispectral satellite data image rectification and orthophoto generation principal and canonical component analysis reclassification and edge detection radiometric correction
visualization	2D display of raster and vector data with zoom and pan 3D visualization of surfaces and volumes with vector data 2D and 3D animations hardcopy postscript maps
modeling and simulations	hydrologic, erosion and pollutant transport, fire spread
temporal data support	time stamp for raster and vector data raster time series analysis
links to Open Source tools	QGIS, R-stats, gstat, UMN/MapServer, Paraview GPS tools, GDAL/OGR, PostgreSQL, MySQL

GRASS project data management

When working with GRASS, it is important to understand that each raster or vector map consists of several files which include the data, categories, header, and other information.

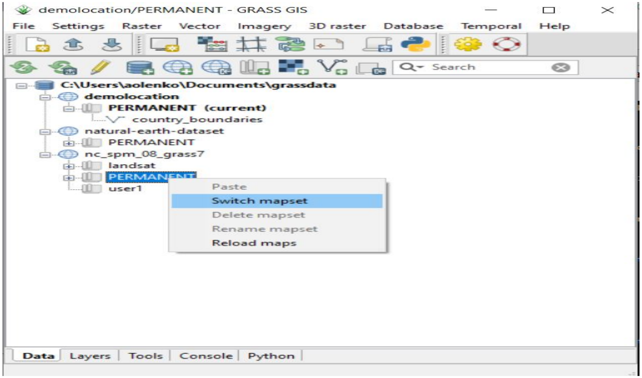
It is not recommended to directly modify the files in the LOCATION or MAPSET directories, unless you are experienced with the system. It is highly recommended to use map management commands that provide listing, copying, renaming and deleting maps. Use them to maintain the consistency in the GRASS GIS directory.

Although it is possible to use all combinations of characters for the GRASS map names if the map name or expression is enclosed within quotes, it is safer to avoid spaces and special characters, such as a comma, dash, or exclamation mark in GRASS map names.

Getting started with GRASS

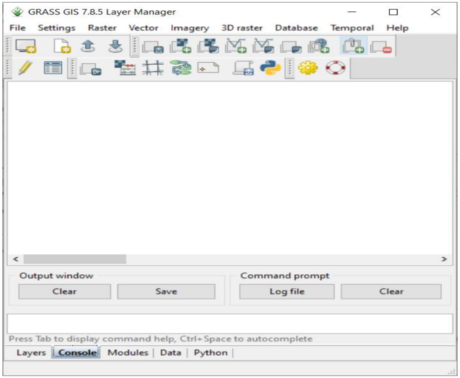
You can call GRASS from Desktop or from the menu system. You will see the start menu and the tab DATA for selecting a LOCATION and a MAPSET in your GIS data directory.

Chose the LOCATION NC_SPM_08_GRASS7 and the MAPSET PERMANENT and click SWITCH MAPSET.



We use both

1. the command line interface, by typing in **Console** in **GRASS GIS Layer Manager** window, and
2. **GUI** interface, which opened when you entered GRASS, use GRASS GIS LAYER MANAGER window's buttons.

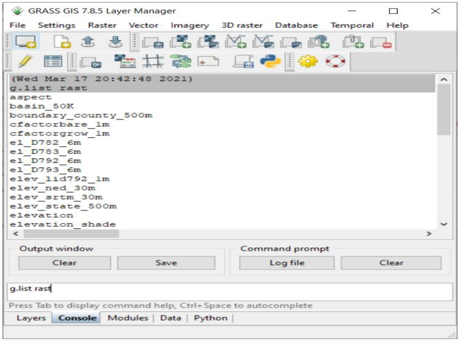

GRASS GIS 7.8.5 Layer Manager
File Settings Raster Vector Imagery 3D raster Database Temporal Help
Output window
Clear
Save
Command prompt
Log file
Clear
Press Tab to display command help, Ctrl+Space to autocomplete
Layers Console Modules Data Python

Command line interface.

Besides the advantage of speed and independence from changing graphical user interfaces, the GRASS shell maintains the command history and auto-documenting per mapset which lets you scroll back.

To list the available raster(vector) maps, type:

```
g.list rast
g.list vect
```


GRASS GIS 7.8.5 Layer Manager
File Settings Raster Vector Imagery 3D raster Database Temporal Help
Output window
Clear Save
Command prompt
Log file Clear
Press Tab to display command help, Ctrl+Space to autocomplete
Layers Console Modules Data Python

```
(Wed Mar 17 20:42:48 2021)
g.list rast
aspect
basin_50K
boundary_county_500m
cfactorbare_1m
cfactorgrow_1m
el_D782_6m
el_D783_6m
el_D792_6m
el_D793_6m
elev_lid792_1m
elev_ned_30m
elev_srtm_30m
elev_state_500m
elevation
elevation_shade
```

```
g.list rast
```

You can learn more about each raster or vector map in terms of its minimum and maximum coordinates, resolution, and number of classes using the *.INFO commands, for example:

```
r.info elevation
v.info streams
v.info -c streams
```

The last command with –c flag prints types and names of attribute table columns for a vector map.

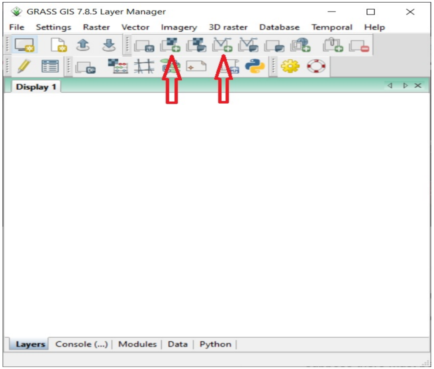
GUI interface GRASS GIS Manager

Most of the GRASS commands are integrated within this interface and you can find a command for a specific task using the function menus. The interface includes a brief description of the parameters and it also displays the command line version of the module.

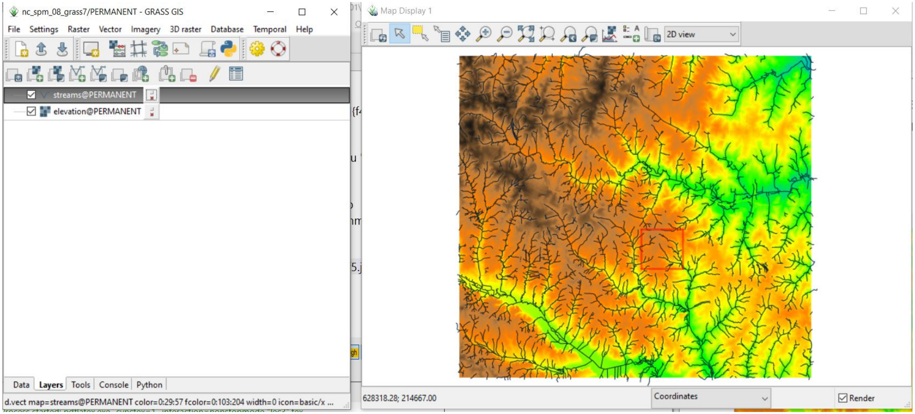
GRASS data display

To view the raster ELEVATION map together with vectors STREAMS (drawn as blue lines) and major roads (drawn as black lines) click the tab Layers and use buttons:

- ADD RASTER MAP LAYER to chose ELEVATION
- ADD VECTOR MAP LAYER to chose STREAMS and ROADSMAJOR



Right mouse click in layer tree and choose from contextual menu "Zoom to selected map", map should appear in canvas in proper size.
You can interactively zoom into a selected location within a map displayed in the GRASS MAP DISPLAY window.



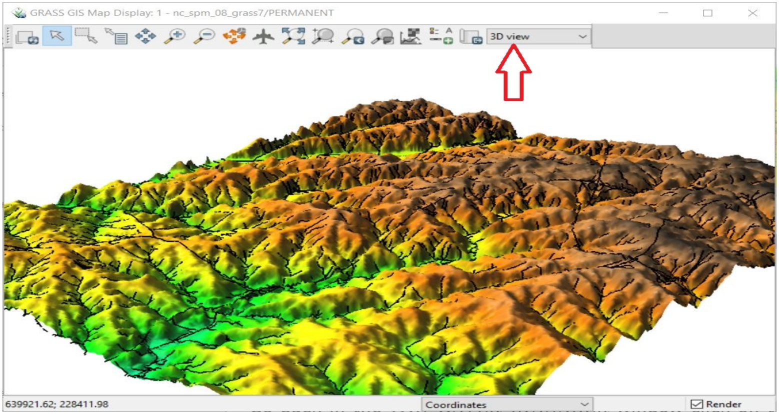
d.vect map=streams@PERMANENT color=0:29:57 fcolor=0:103:204 width=0 icon=basic/x ...

628318.28; 214667.00

To plot 3-d map use the commands:

```
g.region rast=elevation -p
```

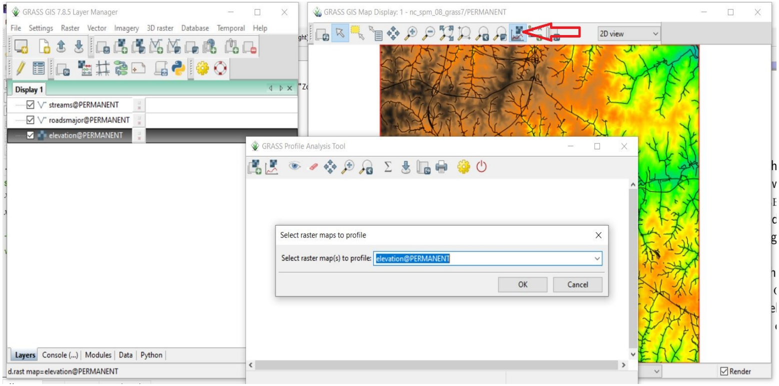
and change 2D View by 3D View in map Display.



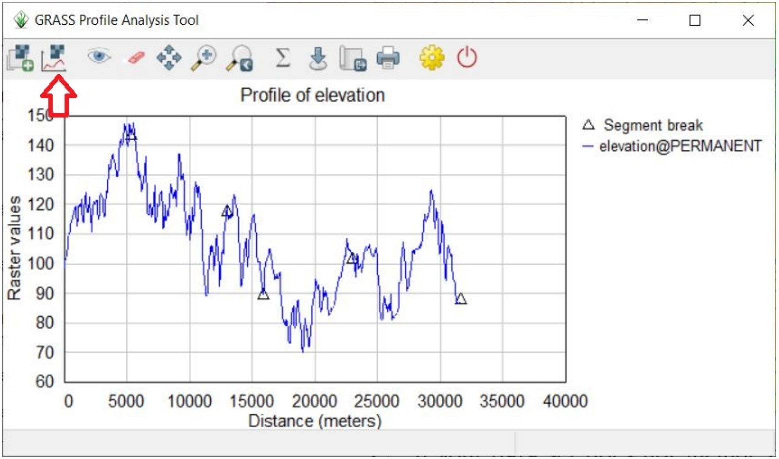
After playing with 3D surface, return to 2D View.

Plot an elevation profile

Go back in the GIS Layer Manager window click on the elevation raster map name to select it. Then in the Map Display window, to the right of the zooming buttons on the Map Display toolbar is an icon with a line graph and checkerboard on it. Click on that and select Profile Surface Map. If it isn't automatically listed again pick an elevation map as the raster layer and press [Ok].


GRASS GIS 7.8.5 Layer Manager
File Settings Raster Vector Imagery 3D raster Database Temporal Help
Display 1
streams@PERMANENT
roadsmajor@PERMANENT
elevation@PERMANENT
Layers Console (...) Modules Data Python
d.rast map=elevation@PERMANENT
GRASS GIS Map Display: 1 - nc_spm_08_grass7/PERMANENT
2D view
Render
GRASS Profile Analysis Tool
Select raster maps to profile
Select raster map(s) to profile: elevation@PERMANENT
OK
Cancel

The second button in from the left allows you to set out the profile line, click it then mark out a few points on the MAP DISPLAY CANVAS. When done go back to the PROFILE window and click on the eyeball button to create the plot. Click on the I/O button of the far right to close the profile window.



Import of georeferenced raster data

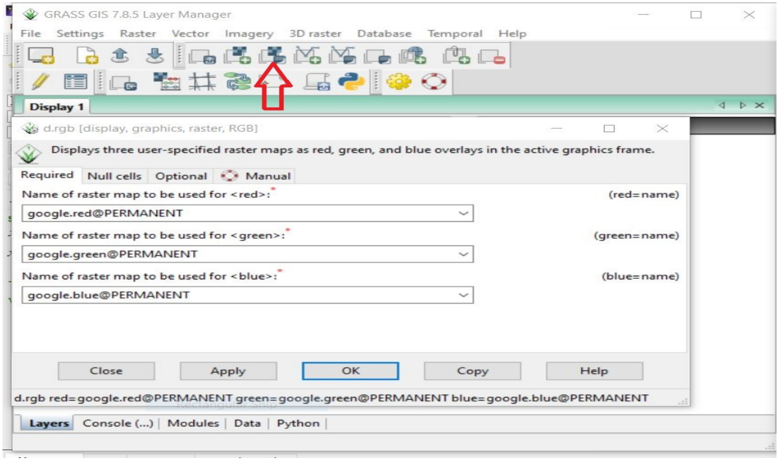
If your data set does not include coordinate system information (projection, datum, and units) check that it matches the coordinate system of your LOCATION.

This is common with data in the TIFF/JPG format. If the imported TIFF image consists of several bands, they are extracted as separate raster maps into the current MAPSET. A typical example is aerial color images delivered in RGB (red, green, blue) channels.

We consider the import a satellite photo from Google Earth. Type in the Console:

```
g.region res=1
r.in.gdal -o D:/google.jpg  out=google
g.region rast=google.blue@PERMANENT
```

To view the resulting raster elevation map use the commands ADD VARIOUS RASTER-BASED MAP LAYERS, ADD RGB MAP LAYER. Right mouse click in layer tree and use "Zoom to selected map" from contextual menu. Map should appear in canvas in proper size.



GRASS GIS 7.8.5 Layer Manager
File Settings Raster Vector Imagery 3D raster Database Temporal Help
Display 1
d.rgb [display, graphics, raster, RGB]
Required Null cells Optional Manual
Close Apply OK Copy Help
Layers Console (...) Modules Data Python

Displays three user-specified raster maps as red, green, and blue overlays in the active graphics frame.

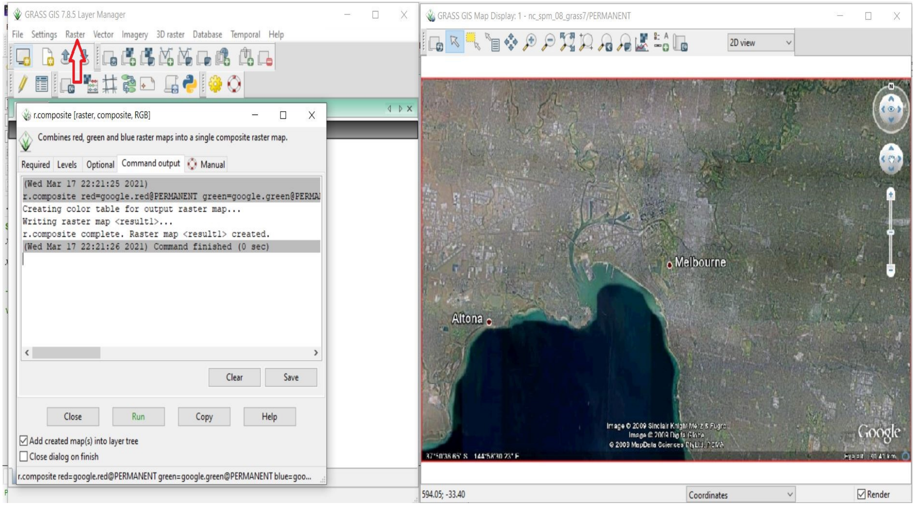
Name of raster map to be used for <red>:* (red=name)
google.red@PERMANENT

Name of raster map to be used for <green>:* (green=name)
google.green@PERMANENT

Name of raster map to be used for <blue>:* (blue=name)
google.blue@PERMANENT

```
d.rgb red=google.red@PERMANENT green=google.green@PERMANENT blue=google.blue@PERMANENT
```

To produce a single raster map from RGB channels use
RASTER -> MANAGE COLORS -> CREATE RGB
with option LEVELS=32.


GRASS GIS 7.8.5 Layer Manager
File Settings Raster Vector Imagery 3D raster Database Temporal Help
r.composite [raster, composite, RGB]
Required Levels Optional Command output Manual
Clear Save
Close Run Copy Help
GRASS GIS Map Display: 1 - nc_spm_08_grass7/PERMANENT
2D view
Melbourne
Altona
Google
Coordinates
Render

Combines red, green and blue raster maps into a single composite raster map.

```
(Wed Mar 17 22:21:25 2021)
r.composite red=google.red@PERMANENT green=google.green@PERMA
Creating color table for output raster map...
Writing raster map <result1>...
r.composite complete. Raster map <result1> created.
(Wed Mar 17 22:21:26 2021) Command finished (0 sec)
```

☑ Add created map(s) into layer tree
☐ Close dialog on finish

r.composite red=google.red@PERMANENT green=google.green@PERMANENT blue=goo...

594.05; -33.40

Automated vectorization of discrete raster data

If the raster data represent linear features, homogeneous areas, or points they can be transformed to vector data.

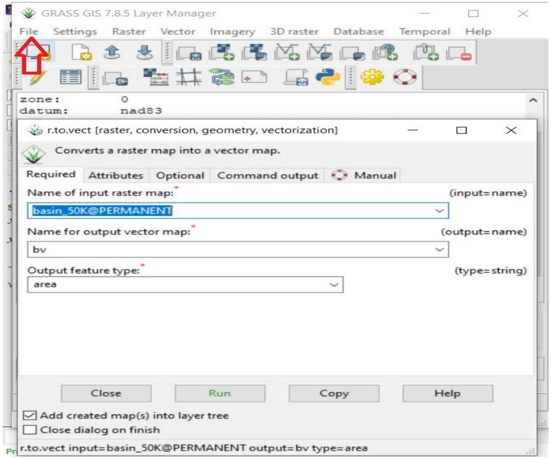
For example, consider vectorizing raster polygons. Using the area borders, we can convert the raster polygons to vector areas. Note that vectorizing areas do not require additional operations and can be done directly. As an example, we vectorize the watershed map BASIN_50K.

Restart GRASS and execute in Console:

```
g.region rast=elevation -p
g.region res=1
```

Then click:
FILE -> MAP TYPE CONVERSION -> RASTER TO VECTOR.

Then chose BASIN_50K, in Output features select AREA, in OPTIONAL tick off SMOOTH CORNERS.... Press RUN.


GRASS GIS 7.8.5 Layer Manager
File Settings Raster Vector Imagery 3D raster Database Temporal Help

```
zone:          0
datum:         nad83
```

r.to.vect [raster, conversion, geometry, vectorization]

Converts a raster map into a vector map.

Required Attributes Optional Command output Manual

Name of input raster map:* (input=name)
basin_50K@PERMANENT

Name for output vector map:* (output=name)
bv

Output feature type:* (type=string)
area

☑ Add created map(s) into layer tree
☐ Close dialog on finish

```
r.to.vect input=basin_50K@PERMANENT output=bv type=area
```

Your can plot the resulting vector map in the GRASS MAP DISPLAY window as before.

Integration of R with GRASS

For the integration of R with GRASS, you need to run R from the GRASS shell environment. The interface dynamically loads compiled GIS library functions into the R environment. The GRASS metadata about the LOCATION's regional extent and raster resolution are transferred to R (internally calling g.region). The current interface is supporting raster and vector data.

Installation of the R/GRASS interface

The installation of the R/GRASS interface is very easy and can be done by a single command (you probably have to be user root for this). If your computer is connected to the Internet, you can install packages within a R session. Start R and launch the command (we install gstat support at the same time):

```
install.packages("rgrass7", "rgdal", dependencies = TRUE)
```

This will download the latest version of the package along with a set of dependency packages; it then unpacks, compiles and installs them.

From time to time, the installed extensions should be checked for updates. The following command will download the R package list, compare to the local installation and upgrade installed packages if new versions are available:

```
update.packages()
```

The path to the R executable depends on your local installation.

Reading GRASS data into R

To illustrate how to apply R to your data, we present examples based on the North Carolina data set.

This data set is a comprehensive collection of raster, vector and imagery data covering parts of North Carolina (NC), USA.

Data are provided at three hierarchical levels:

- entire NC with raster data at 500m resolution (boundary in geographic coordinates: 37N-33N,75W-85W);
- SouthwestWake county with raster data at resolutions 30m-10m (boundary coordinates 35:48:34.6N-35:41:15.0N, 78:46:28.6W-78:36:29.9W), and
- a small watershed in rural area with data resolutions of 1m-3m. The data set includes section of the NC capital city Raleigh and its surroundings.

The coordinate system of the ready-to-use GRASS data set is NC State Plane (Lambert Conformal Conic projection) and metric units.

Additional data are provided in geographic coordinates and NC State Plane, english units (feet) in various external formats.

Vector data include:
administrative boundaries, census data, zipcodes, firestations, hospitals, roads and railroads, public schools and colleges, bus routes, points of interest, precipitation, hydrography maps, geodetic points, soils and geological maps.

Raster data include:
elevation , slope, watershed basins, geology, landuse, etc.

More complete data description can be found at the GRASS book site.

Example 1.

As an exercise, we will analyze precipitation normals. The normal precipitation is the arithmetic mean for each month over the *n* year period.

After starting GRASS with North Carolina nc_spm LOCATION and setting the region to the precipitation map (along with 1000m resolution), we launch R within the GRASS terminal:

```
g.region vect=precip_30ynormals res=1000 -ap
C:\Users\aolenko\Documents\R\...\bin\x64\Rgui
```

If you need to change the region settings later within R, you can use the SYSTEM() function to call G.REGION.

Within R the R/GRASS interface is loaded as follows:

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

By this, we have loaded the library of interface functions and loaded GRASS metadata about the location into the R environment.

Now we are ready to perform geospatial analysis of GRASS raster and vector data. As a prerequisite, to process these spatial data in R, we import some maps into the R environment.

To start, we load the North Carolina 30 year monthly and annual precipitation normals precip_30ynormals and the vector map nc_state representing the state political boundaries:

```r
precip30n <- readVECT("precip_30ynormals",ignore.stderr=TRUE)
nc_state <- readVECT("nc_state", ignore.stderr=TRUE)
```

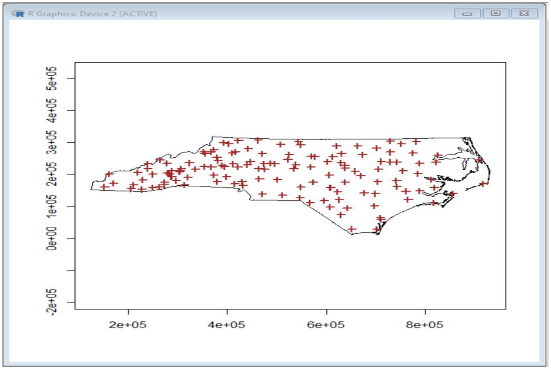
Then we verify that it is a SpatialPointsDataFrame:

```
class(precip30n)
summary(precip30n)
```

The SUMMARY() function computes the summary characteristics for the precipitation data set including univariate statistics (minimum, maximum, 1st and 3rd quartile, median, mean and the number of NAs).

Since we are working with spatial data, we can plot the R object PRECIP30N as a simple point map with the NC state map as background:

```
plot(nc_state, axes=TRUE)
plot(precip30n, add=TRUE, lwd=2, col="brown")
```



This map only shows the locations of the meteorological stations in North Carolina available in our data set.

To see all data objects which are currently loaded into R, use the $\mathrm{LS}()$ function. Next we list the variables in the R object:

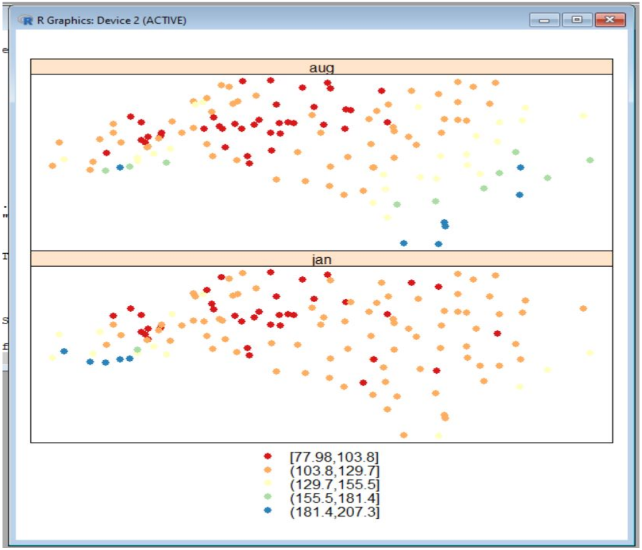
```
> ls()
[1] "CRAN_df"    "CRAN_mat"   "CRAN_sp"    "CRAN_spdf1"
[5] "llCRS"      "nc_state"   "precip30n"
> names(precip30n)
[1] "cat"     "station" "lat"     "long"    "elev"
[6] "jan"     "feb"     "mar"     "apr"     "may"
[11] "jun"     "jul"     "aug"     "sep"     "oct"
[16] "nov"     "dec"     "annual"
```

To produce spatial plot of selected months, load color library first:

```
library(RColorBrewer)
```

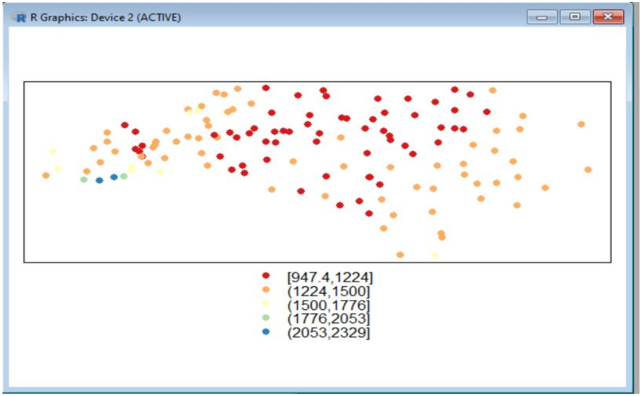
Spatial plot of map in two monthly lattice panels:

```
library(RColorBrewer)
spplot(precip30n, c("jan", "aug"),
+ col.regions=brewer.pal(5,"Spectral"))
```


R Graphics: Device 2 (ACTIVE)
aug
jan
● [77.98,103.8]
● (103.8,129.7]
● (129.7,155.5]
● (155.5,181.4]
● (181.4,207.3]

Annual precipitation:

```
spplot(precip30n, "annual",
+ col.regions=brewer.pal(5,"Spectral"))
```



The two plotted maps show different patterns in winter and summer precipitation. The annual precipitation shows average precipitation along the costline and higher precipitation in the South-West NC.

Example 2.

Now we study how to import maps from R into the GRASS environment. We consider our example of SpatialPointsDataFrame with the positions of CRAN mirrors across the world:

```
library(rgrass7)
library(rgdal)
library(sp)
CRAN_df <- read.table("D:/CRAN051001a.txt", header = TRUE)
CRAN_mat <- cbind(CRAN_df$long, CRAN_df$lat)
row.names(CRAN_mat) <- 1:nrow(CRAN_mat)
str(CRAN_mat)
llCRS <- CRS("+proj=longlat +ellps=WGS84")
CRAN_sp <- SpatialPoints(CRAN_mat, proj4string = llCRS)
summary(CRAN_sp)
CRAN_spdf1 <- SpatialPointsDataFrame(CRAN_mat,
+CRAN_df,proj4string = llCRS, match.ID = TRUE)
```

Find help on "write to GRASS" by using

```
?readVECT
```

Then, to save the map in GRASS as NEWCRAN, use:

```
writeVECT(CRAN_spdf1,"newcran",
+ v.in.ogr_flags=c("o", "overwrite"),
+ ignore.stderr=TRUE)
```

Now one can plot and modify the vector map NEWCRAN in GRASS as usual.

If one wants to call GRASS within R and obtain information about the new map, they can use

```
 execGRASS("v.info", parameters=list(map="newcran"))
```

```
+----------------------------------------------------------------------------+
| Name:             newcran                                                  |
| Mapset:           PERMANENT                                                |
| Location:         nc_spm_08_grass7                                         |
| Database:         C:\Andriy\STA4SA\GISDATABASE                             |
| Title:                                                                     |
| Map scale:        1:1                                                      |
| Name of creator: AOlenko                                                   |
| Organization:                                                              |
| Source date:      Wed Apr 14 11:37:39 2021                                 |
| Timestamp (first layer): none                                              |
|----------------------------------------------------------------------------|
| Map format:       native                                                   |
|----------------------------------------------------------------------------|
|   Type of map: vector (level: 2)                                           |
|                                                                            |
|   Number of points:       54              Number of centroids:  0          |
|   Number of lines:        0               Number of boundaries: 0          |
|   Number of areas:        0               Number of islands:    0          |
|                                                                            |
|   Map is 3D:              No                                               |
|   Number of dblinks:      1                                                |
|                                                                            |
|   Projection: Lambert Conformal Conic                                      |
|                                                                            |
|               N:              57.05    S:      -37.81666667               |
|               E:       153.03333333    W:           -122.95               |
|                                                                            |
|   Digitization threshold: 0                                                |
|   Comment:                                                                 |
|                                                                            |
+----------------------------------------------------------------------------+
```

Key R commands

library(rgrass7)	*interface between GRASS geographical information system and R*
library(rgdal)	*provides bindings to the 'Geospatial' Data Abstraction Library*
use_sp()	*reads GRASS metadata from the current LOCATION*
readVECT(x)	*reads a GRASS vector object to a Spatial*DataFrame object*
writeVECT(x)	*writes "SpatVector" object to a GRASS vector object*
execGRASS(x)	*runs GRASS commands in R*
library(RColorBrewer)	*provides color schemes for maps*