Mapping

Mapping

Introduction to MATLAB mapping Toolbox

Map projection

MATLAB mapping Toolbox and georeferencing

Laboratory exercises:

Romà Tauler (IDAEA, CSIC, Barcelona)

Maps

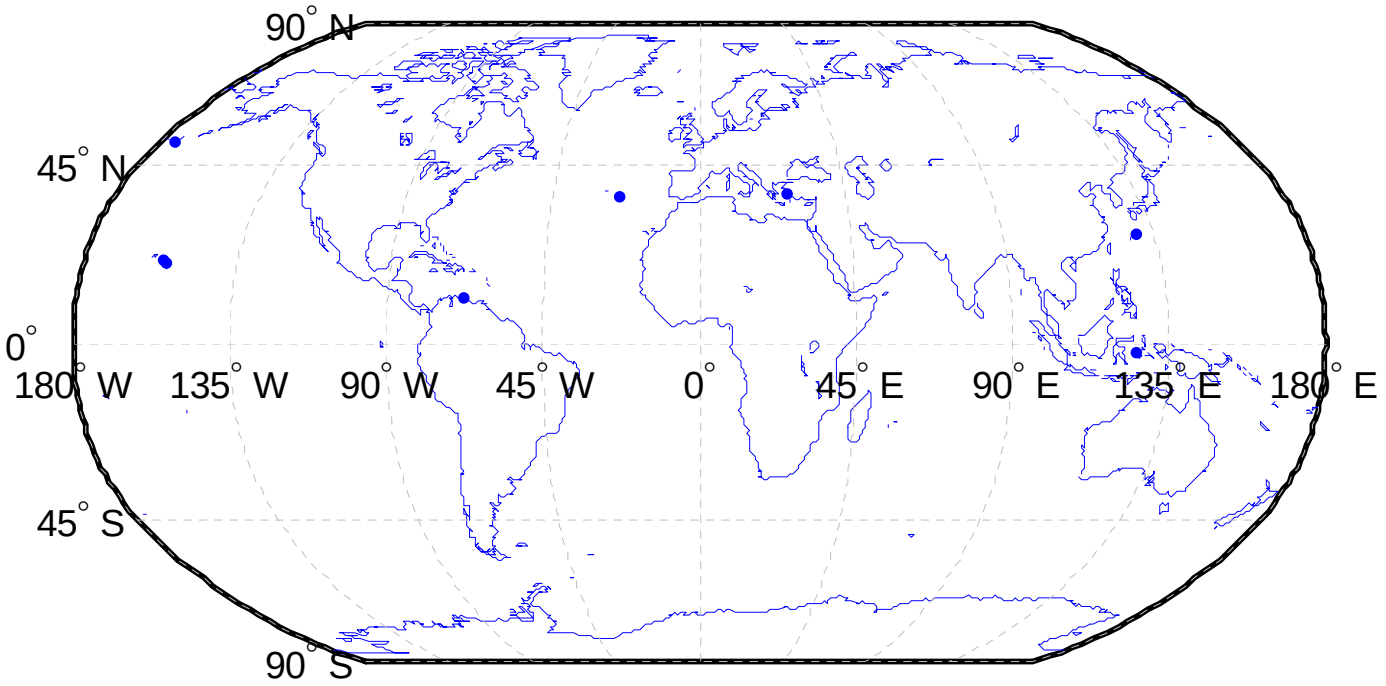
The simplest (although perhaps not the most general) definition of a map is a representation of geographic data.

Map data is any variable or set of variables representing a set of **geographic locations**, properties of a region, or features on a planet's surface, regardless of how large or complex the data is, or how it is formatted.

Geospatial data are spatial data (data where things are within a given coordinate system) that are absolutely or relatively positioned on a planet, or **georeferenced** (it has a terrestrial coordinate system that can be shared by other geospatial data)

Geodata is coded for computer storage and applications in two principal ways: **vector and raster representations**. It has been said that "raster is faster but vector is corrector

Map examples

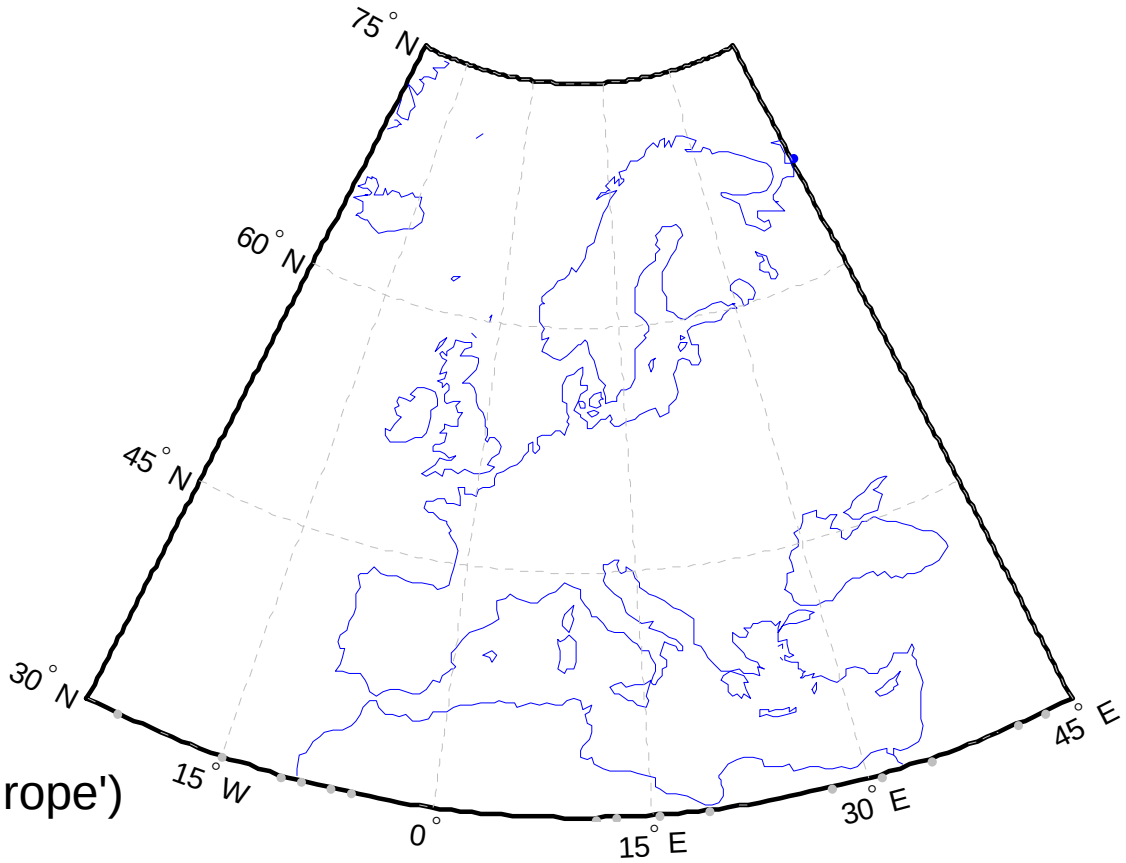


Example 1

```
% World map with coarse coastlines
worldmap('World')
load coast
plotm(lat, long)
```

World

Map examples



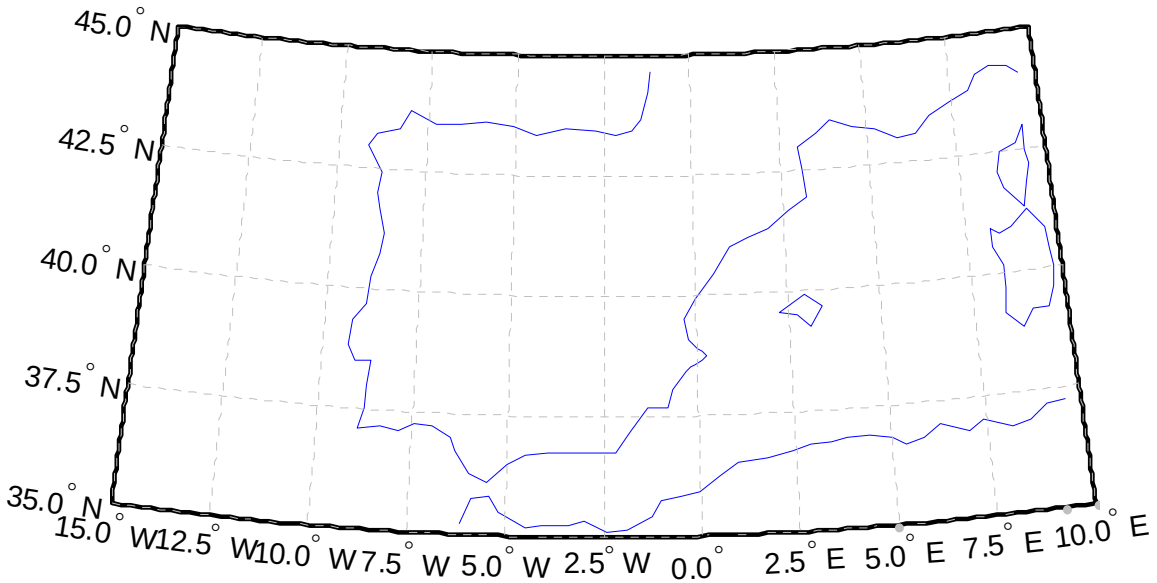
worldmap('Europe')

load coast

plotm(lat, long)

Europe

Map examples



worldmap('Spain')

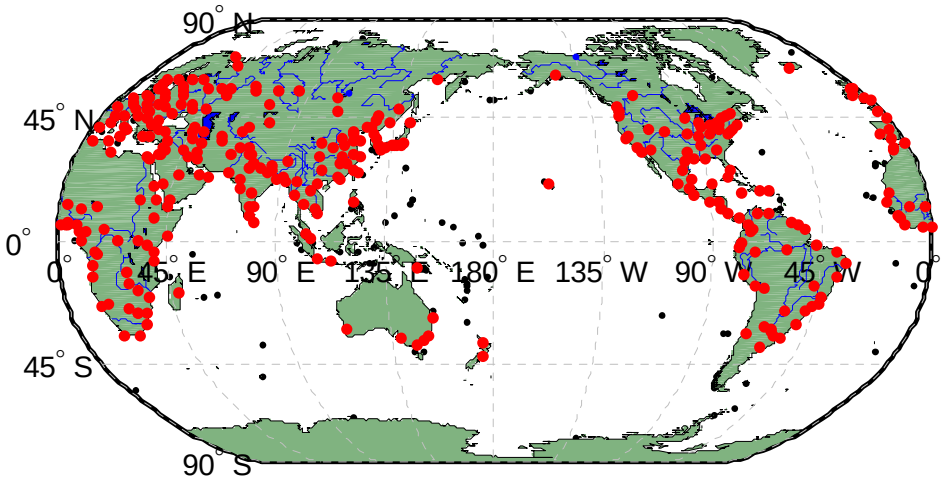
load coast

plotm(lat, long)

Spain

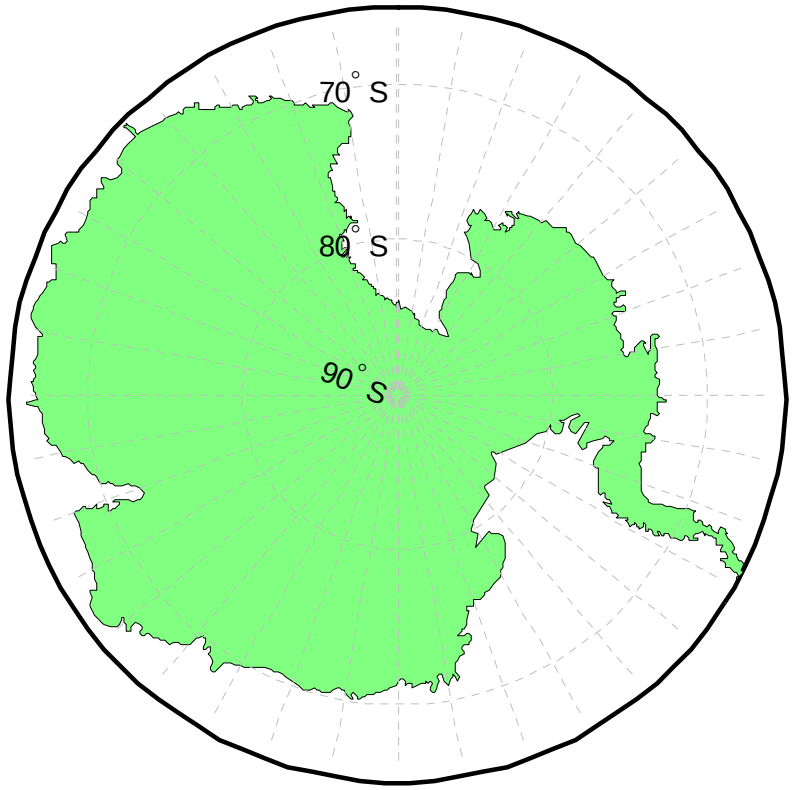
Example 2

```
% Worldmap with land areas, major lakes and rivers, and cities and
% populated places
ax = worldmap('World');
setm(ax, 'Origin', [0 180 0])
land = shaperead('landareas', 'UseGeoCoords', true);
geoshow(ax, land, 'FaceColor', [0.5 0.7 0.5])
lakes = shaperead('worldlakes', 'UseGeoCoords', true);
geoshow(lakes, 'FaceColor', 'blue')
rivers = shaperead('worldrivers', 'UseGeoCoords', true);
geoshow(rivers, 'Color', 'blue')
cities = shaperead('worldcities', 'UseGeoCoords', true);
geoshow(cities, 'Marker', '.', 'Color', 'red')
```



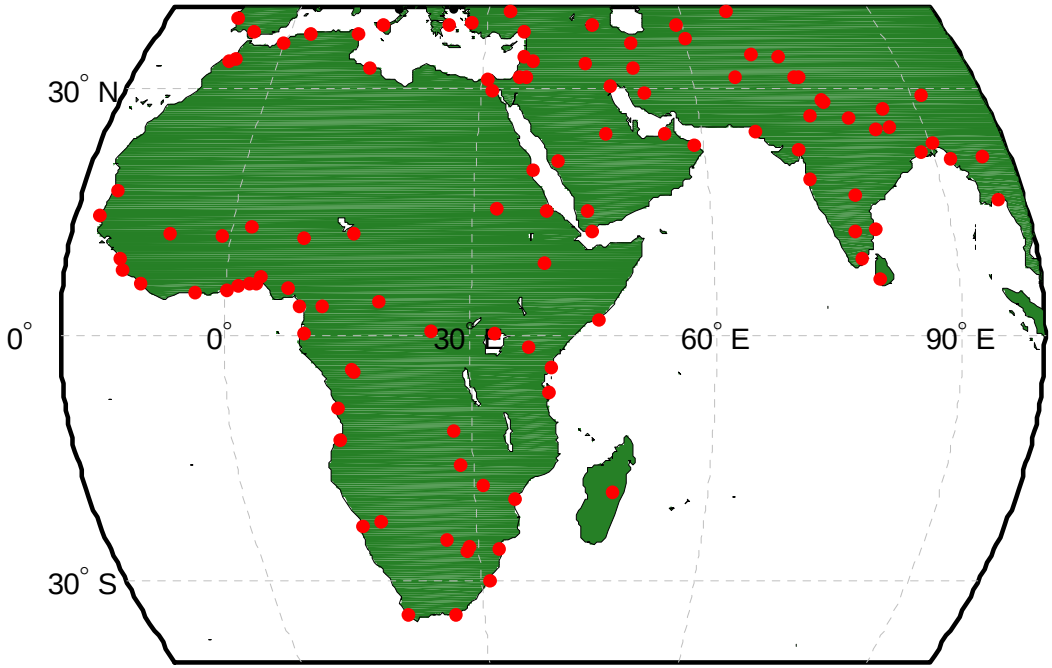
Example 3

```
---------
% Map of Antarctica
worldmap('antarctica')
antarctica = shaperead('landareas', 'UseGeoCoords', true,...
    'Selector',{@(name) strcmp(name,'Antarctica'), 'Name'});
patchm(antarctica.Lat, antarctica.Lon, [0.5 1 0.5])
```



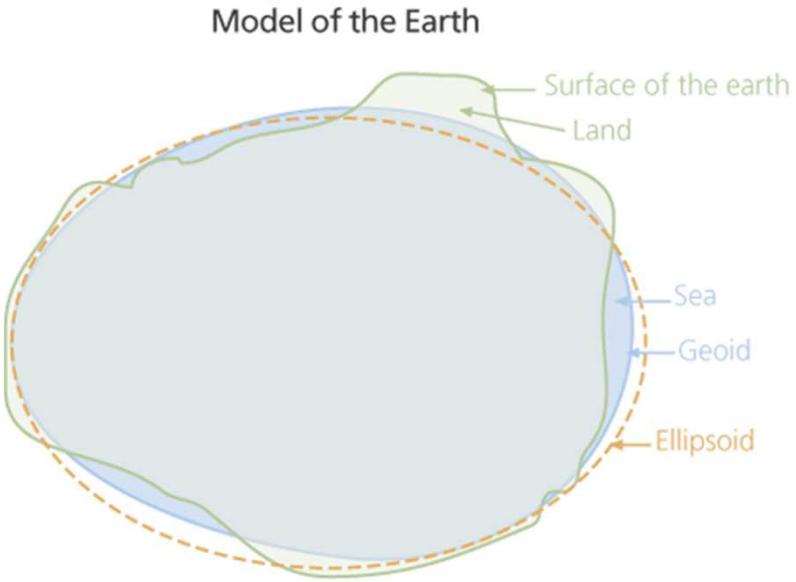
Example 4

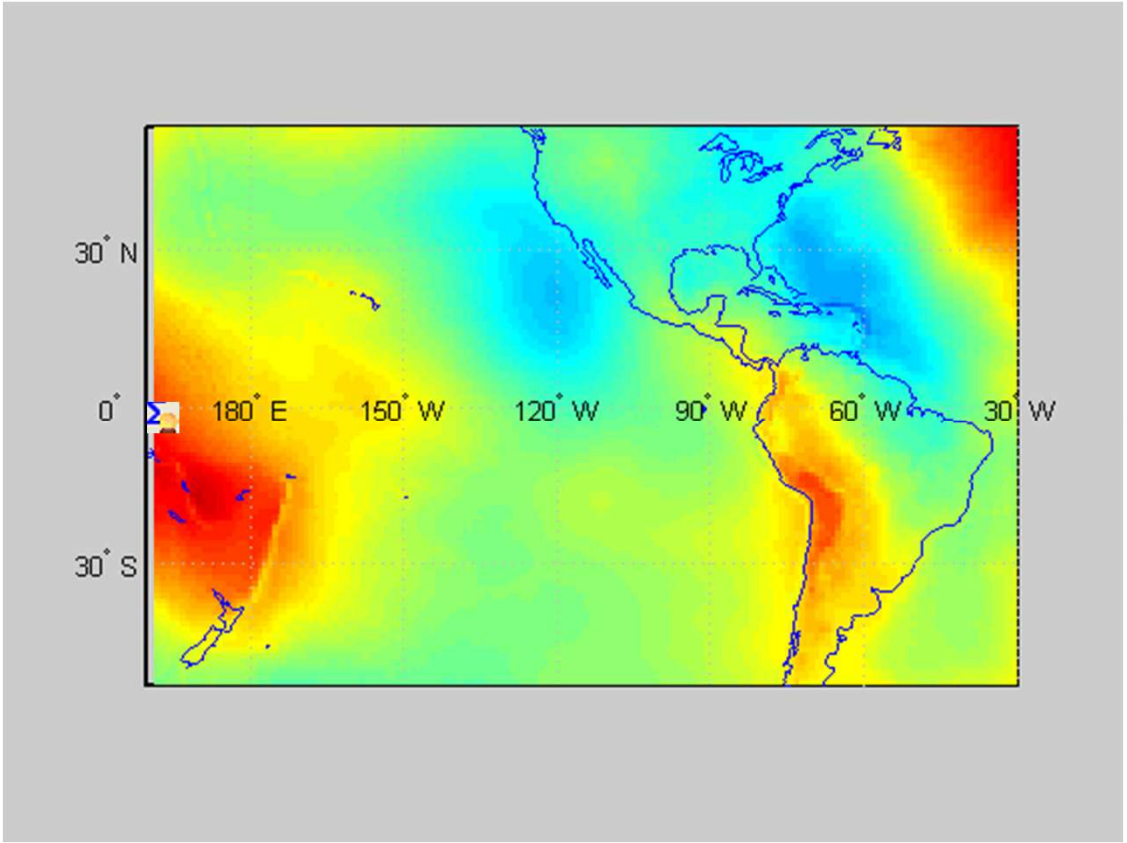
```
% Map of Africa and India with major cities and populated places
worldmap({'Africa','India'})
land = shaperead('landareas.shp', 'UseGeoCoords', true);
geoshow(land, 'FaceColor', [0.15 0.5 0.15])
cities = shaperead('worldcities', 'UseGeoCoords', true);
geoshow(cities, 'Marker', '.', 'Color', 'blue')
```



Example 5 ---------

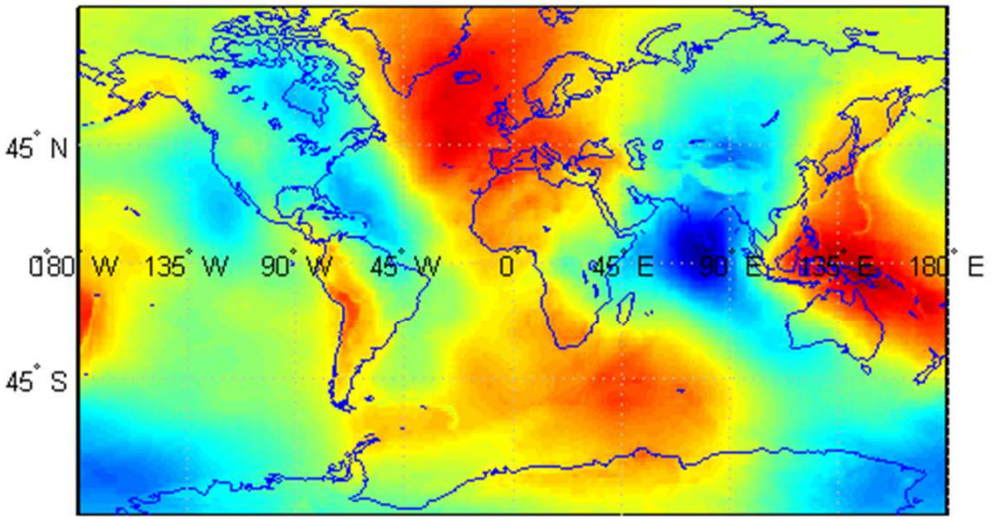
```
% Map of the geoid over South America and the central Pacific
worldmap([-50 50],[160 -30])
load geoid
geoshow(geoid, geoidrefvec, 'DisplayType', 'texturemap');
load coast
geoshow(lat, long)
```

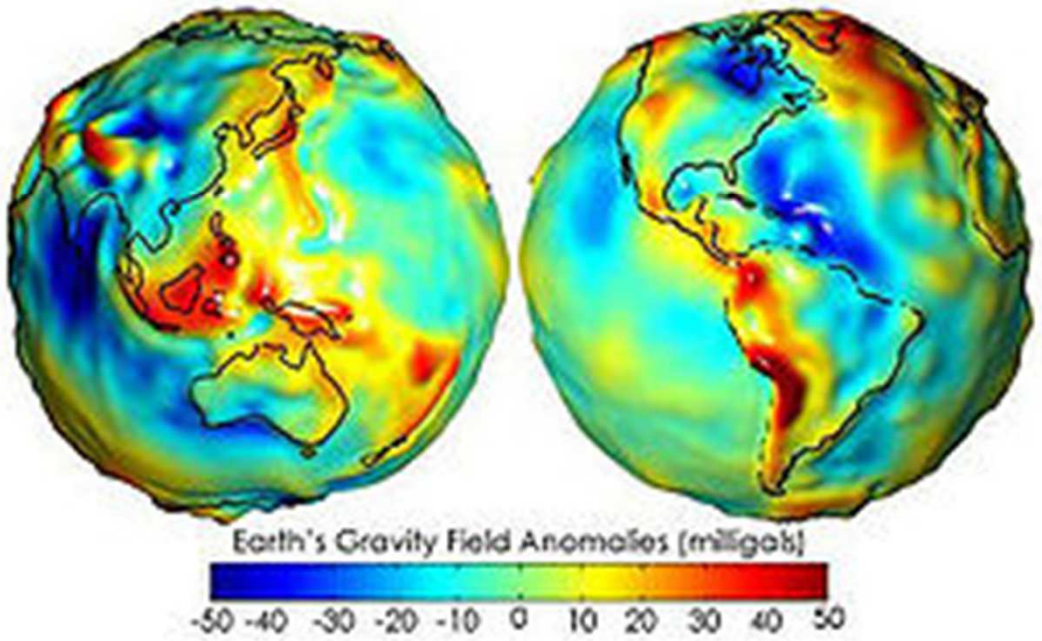




The geoid approximates mean sea level. The shape of the ellipsoid was calculated based on the hypothetical equipotential gravitational surface. A significant difference exists between this mathematical model and the real object. However, even the most mathematically sophisticated geoid can only approximate the real shape of the earth.

geoid, model of the figure of the Earth—i.e., of the planet's size and shape—that coincides with mean sea level over the oceans and continues in continental areas as an imaginary sea-level surface defined by spirit level. It serves as a reference surface from which topographic heights and ocean depths are measured. The scientific discipline concerned with the precise figure of the Earth and its determination and significance is known as geodesy.





Example 6

```
% Map of terrain elevations in Korea
load korea
h = worldmap(map, refvec);
set(h, 'Visible', 'off')
geoshow(h, map, refvec, 'DisplayType', 'texturemap')
colormap(demcmap(map))
```

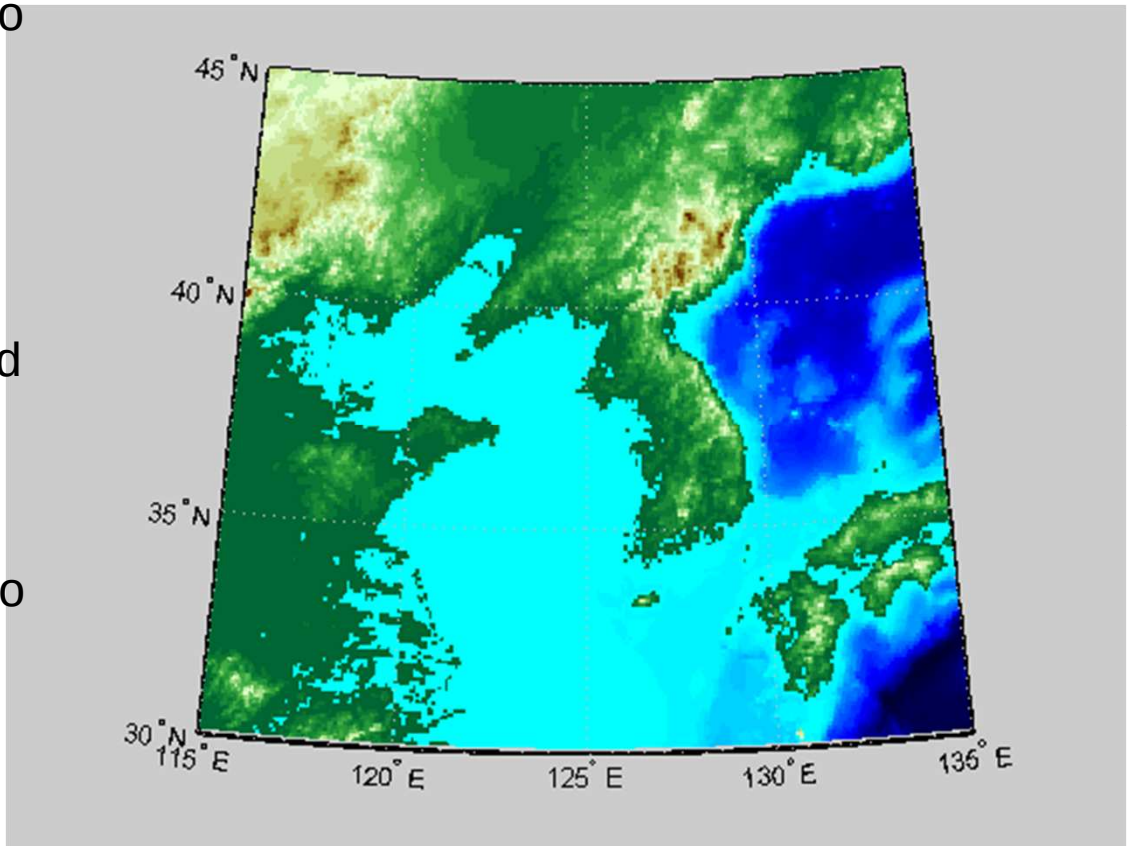
not display the outside of the projection

DEMCMAP Colormaps appropriate to terrain elevation data

DEMCMAP(map) creates and assigns a colormap appropriate for elevation data.

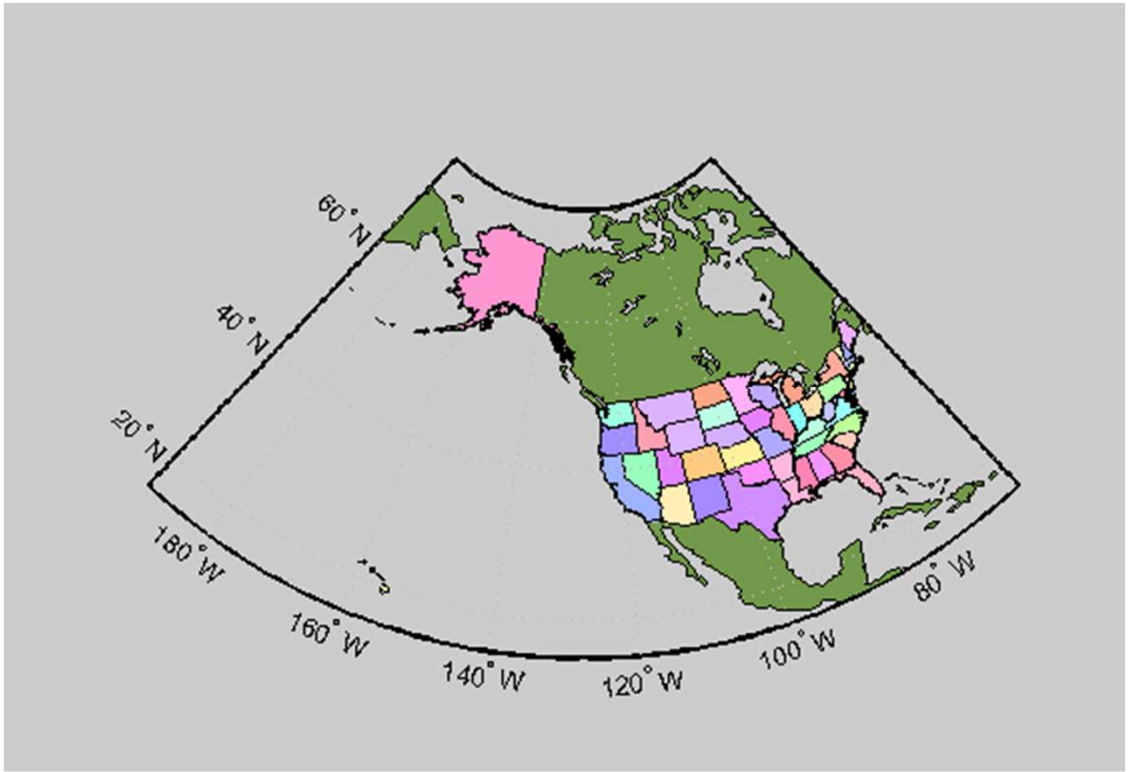
The colormap has the number of land and sea colors in proportion to the maximum elevations and depths in the matrix map. With no output arguments, the colormap is applied to the current figure and the axes

CLim property is set so that the interface between the land and sea is correct.



Example 7

```
---------
% Map of the United States of America
ax = worldmap('USA');
load coast
geoshow(ax, lat, long,...
    'DisplayType', 'polygon', 'FaceColor', [.45 .60 .30])
states = shaperead('usastatelo', 'UseGeoCoords', true);
faceColors = makesymbolspec('Polygon',...
    {'INDEX', [1 numel(states)], 'FaceColor', polcmap(numel(states))});
geoshow(ax, states, 'DisplayType', 'polygon', 'SymbolSpec', faceColors)
```



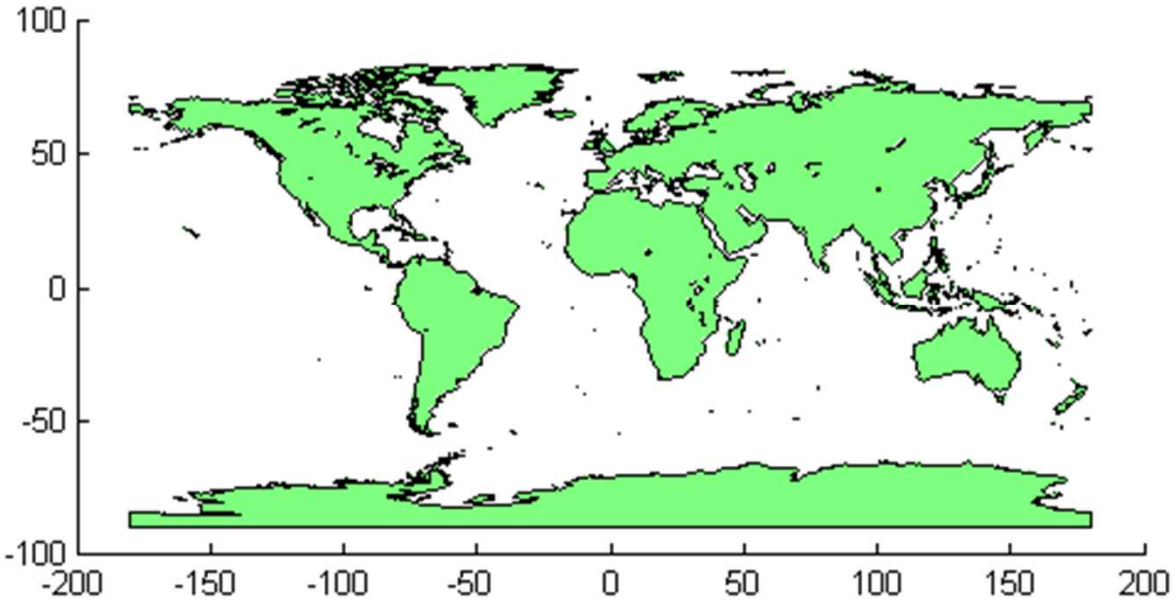
GEOSHOW Display map latitude and longitude data

GEOSHOW(LAT, LON) or
GEOSHOW(LAT, LON, ..., 'DisplayType', DISPLAYTYPE, ...)
GEOSHOW(LAT,LON,Z, ..., 'DisplayType', DISPLAYTYPE, ...)
GEOSHOW(Z,R, ..., 'DisplayType', DISPLAYTYPE,...)
GEOSHOW(LAT,LON,I)
GEOSHOW(LAT,LON,BW)
GEOSHOW(LAT,LON,X,CMAP)
GEOSHOW(LAT,LON,RGB)
GEOSHOW(..., 'DisplayType', DISPLAYTYPE, ...)
GEOSHOW(I,R)
GEOSHOW(BW,R)
GEOSHOW(RGB,R)
GEOSHOW(A,CMAP,R)
GEOSHOW(..., 'DisplayType', DISPLAYTYPE, ...)
GEOSHOW(S) or GEOSHOW(S, ..., 'SymbolSpec', SYMSPEC, ...)
GEOSHOW(FILENAME)
.....

Look at HELP GEOSHOW, a very powerful MATLAB command

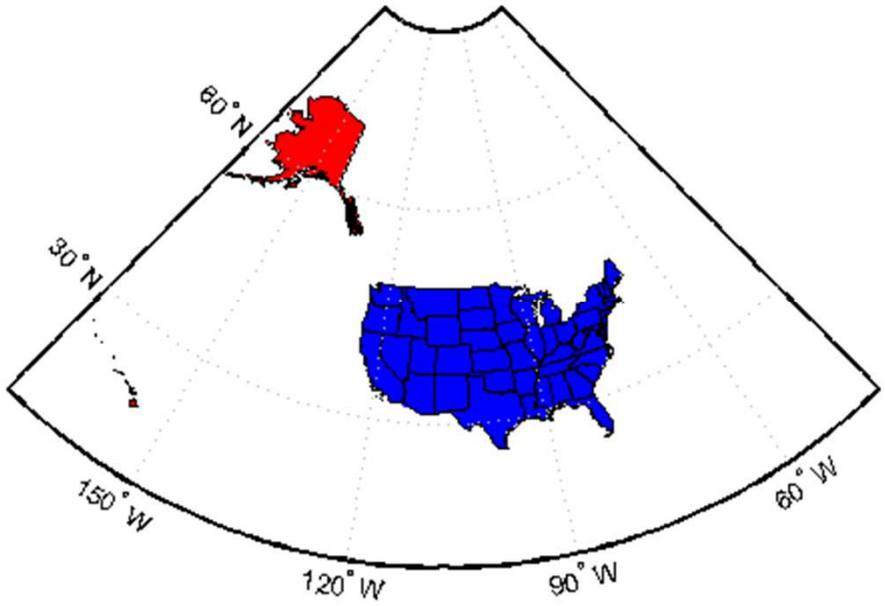
Example 1

```
---------
% Display world land areas using a default Plate Carree
projection.
figure
geoshow('landareas.shp', 'FaceColor', [0.5 1.0 0.5]);
```



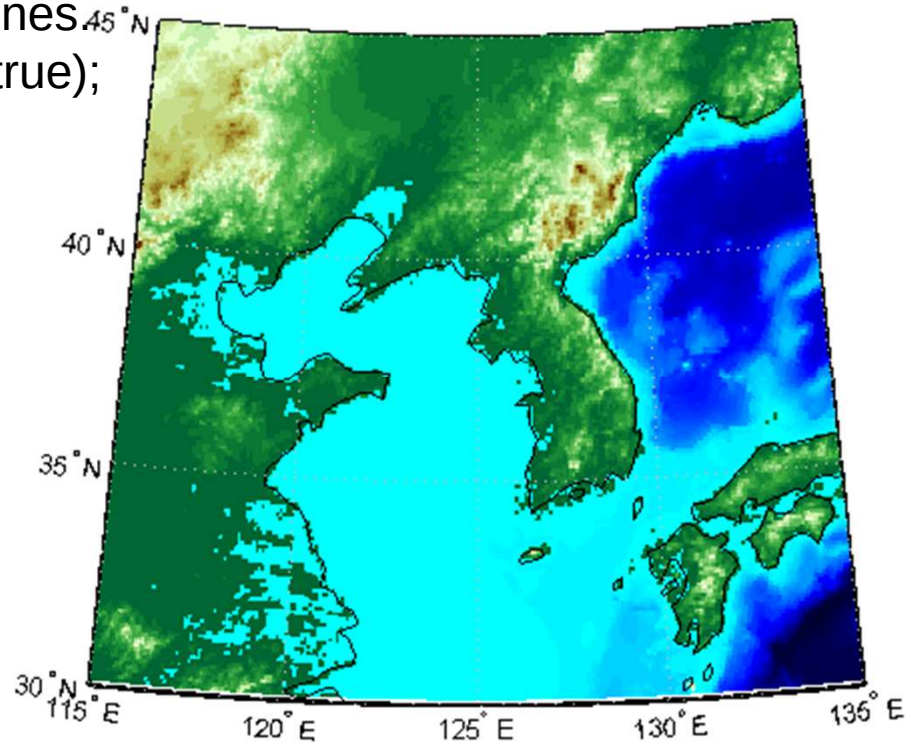
Example 2

```
% Create a worldmap of North America, and
% override default polygon properties.
figure
worldmap('na');
% Read the USA high resolution data.
states = shaperead('usastatehi', 'UseGeoCoords', true);
% Create a SymbolSpec to display Alaska and Hawaii as red polygons.
symbols = makesymbolspec('Polygon', ...
                         {'Name', 'Alaska', 'FaceColor', 'red'}, ...
                         {'Name', 'Hawaii', 'FaceColor', 'red'});
% Display all the other states in blue.
geoshow(states, 'SymbolSpec', symbols, ...
                'DefaultFaceColor', 'blue', ...
                'DefaultEdgeColor', 'black');
```



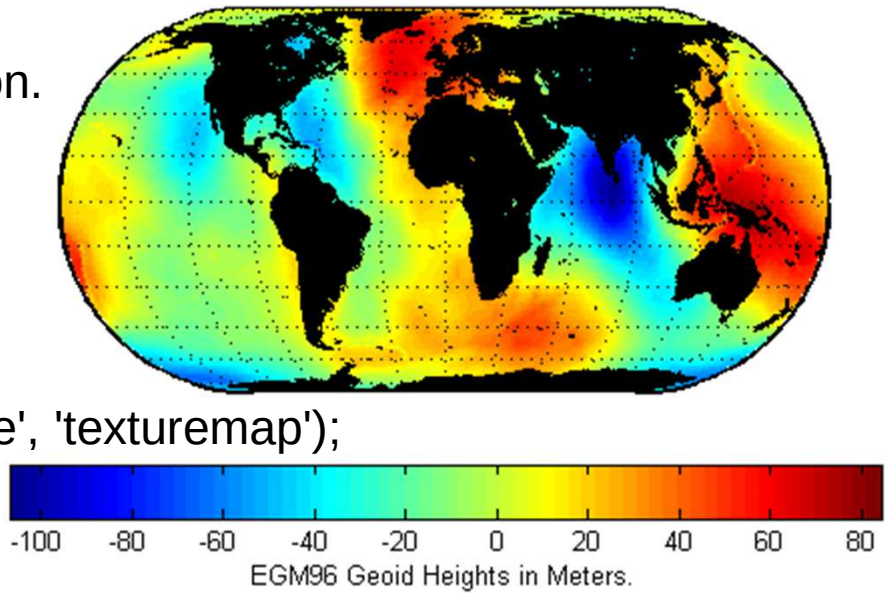
Example 3

```
% Create a worldmap of the Korean data grid
% and display as a texture map.
load korea
figure
worldmap(map, refvec)
% Display the Korean data grid as a texture map.
geoshow(gca,map,refvec,'DisplayType','texturemap');
colormap(demcmap(map))
% Display the land area boundary as black lines
S = shaperead('landareas','UseGeoCoords',true);
geoshow([S.Lat], [S.Lon], 'Color' ,'black');
```



Example 4

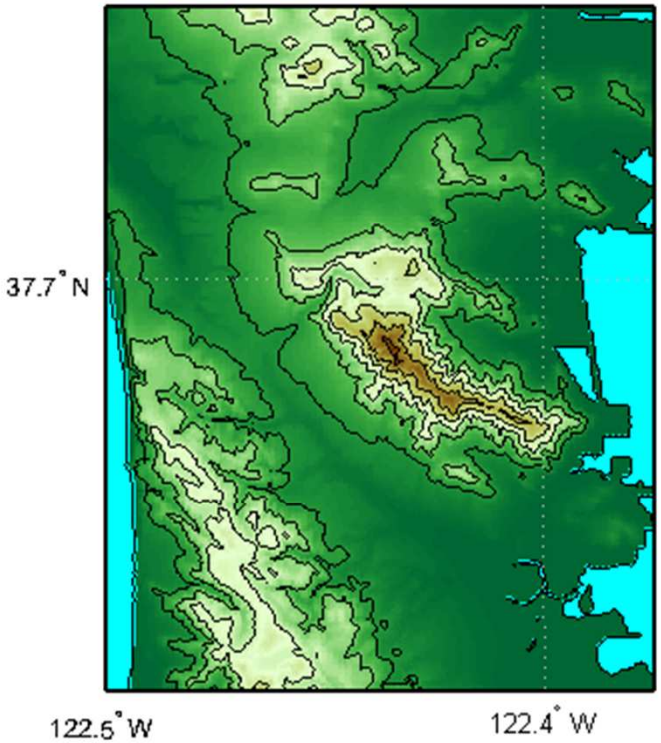
```
% Display the EGM96 geoid heights as a   texture map
% using the Eckert projection.
load geoid
% Create a figure with an Eckert projection.
figure
axesm eckert4;
framem; gridm;
axis off
% Display the geoid as a texture map.
geoshow(geoid, geoidrefvec, 'DisplayType', 'texturemap');
% Create a colorbar and title.
hcb = colorbar('horiz');

set(get(hcb,'Xlabel'),'String','EGM96 Geoid Heights in Meters.')
% Mask out all the land.
geoshow('landareas.shp', 'FaceColor', 'black');
```

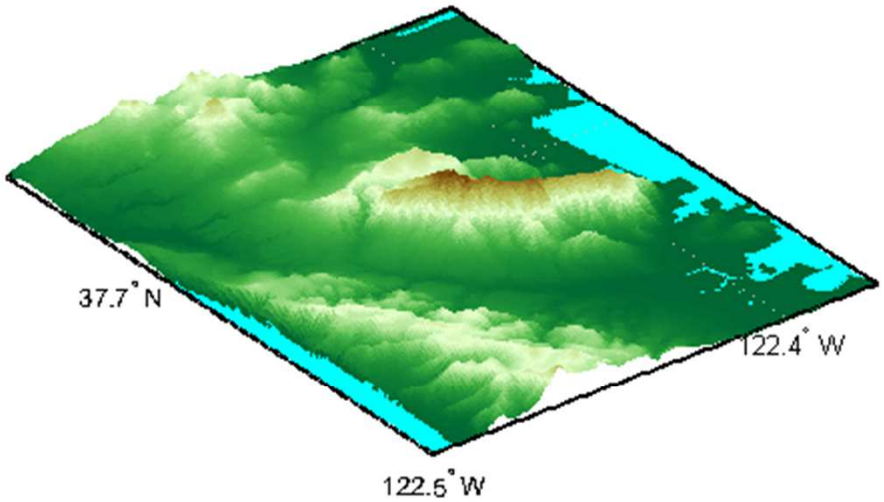


Example 6

```
---------
% Display the moon albedo image using a default Plate Carree projection
% and an orthographic projection.
load moonalb
% Plate Carree projection.
figure
geoshow(moonalb,moonalbrefvec)
% Orthographic projection.
figure
axesm ortho
geoshow(moonalb, moonalbrefvec, 'DisplayType', 'texturemap')
colormap(gray(256))
axis off
```

```
% Example 7  Read and display the San Francisco South 24K DEM data.
    filenames = gunzip('sanfranciscos.dem.gz', tempdir);
    demFilename = filenames{1};
     % Read every point of the 1:24,000 DEM file.
    [lat, lon,Z] = usgs24kdem(demFilename,2);
     % Delete the temporary gunzipped file.
    delete(demFilename);
     % Move all points at sea level to -1 to color them blue.
    Z(Z==0) = -1;
     % Compute the latitude and longitude limits for the DEM.
    latlim = [min(lat(:)) max(lat(:))];
    lonlim = [min(lon(:)) max(lon(:))];
    % Display the DEM values as a texture map.
    figure
    usamap(latlim, lonlim)
    geoshow(lat, lon, Z, 'DisplayType','texturemap')
    demcmap(Z)
    daspectm('m',1)
    % Overlay black contour lines onto the texturemap.
    geoshow(lat, lon, Z, 'DisplayType', 'contour', 'LineColor', 'black');
    % View the DEM values in 3-D.
    figure
    usamap(latlim, lonlim)
    geoshow(lat, lon, Z, 'DisplayType', 'surface')
    demcmap(Z)
    daspectm('m',1)
    view(3)
```


37.7° N
122.5° W
122.4° W


37.7° N
122.4° W
122.5° W

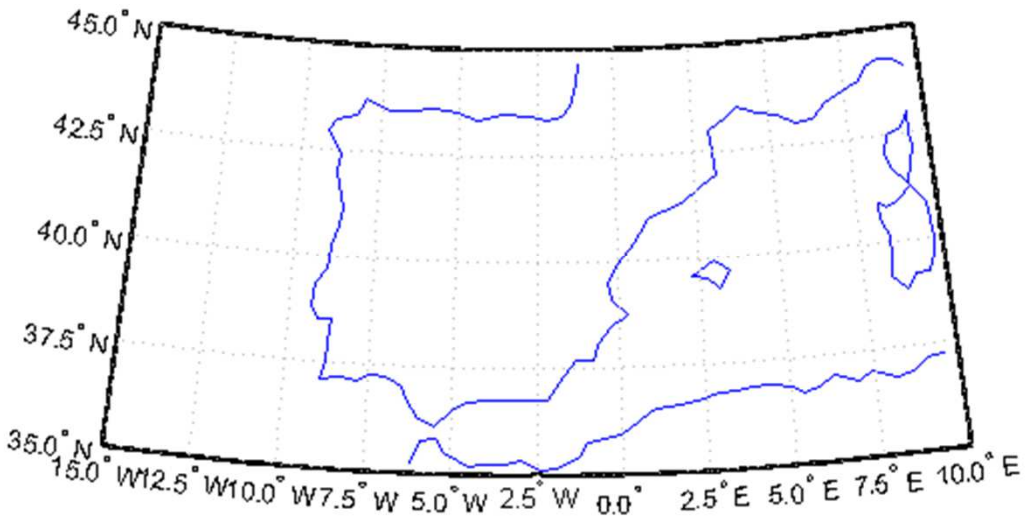
Other Examples

Try: worldmap command (mapping Toolbox) and select region...

>> worldmap ---> Spain or worldmap('Spain')

>> load coast

>> plotm(lat,long)



Geographic coordinates

Map coordinate systems - geographic and geomagnetic.

Latitude/Longitude is the usual coordinate system for maps.

In some cases UTM coords are also used, but these are really just a simple transformation based on the location of the equator and certain lines of longitude.

On the other hand, there are occasions when a coordinate system based on some other set of axes is useful. For example, in space physics data is often projected in coordinates based on the magnetic poles

Notice that longitudes are specified using a *signed* notation

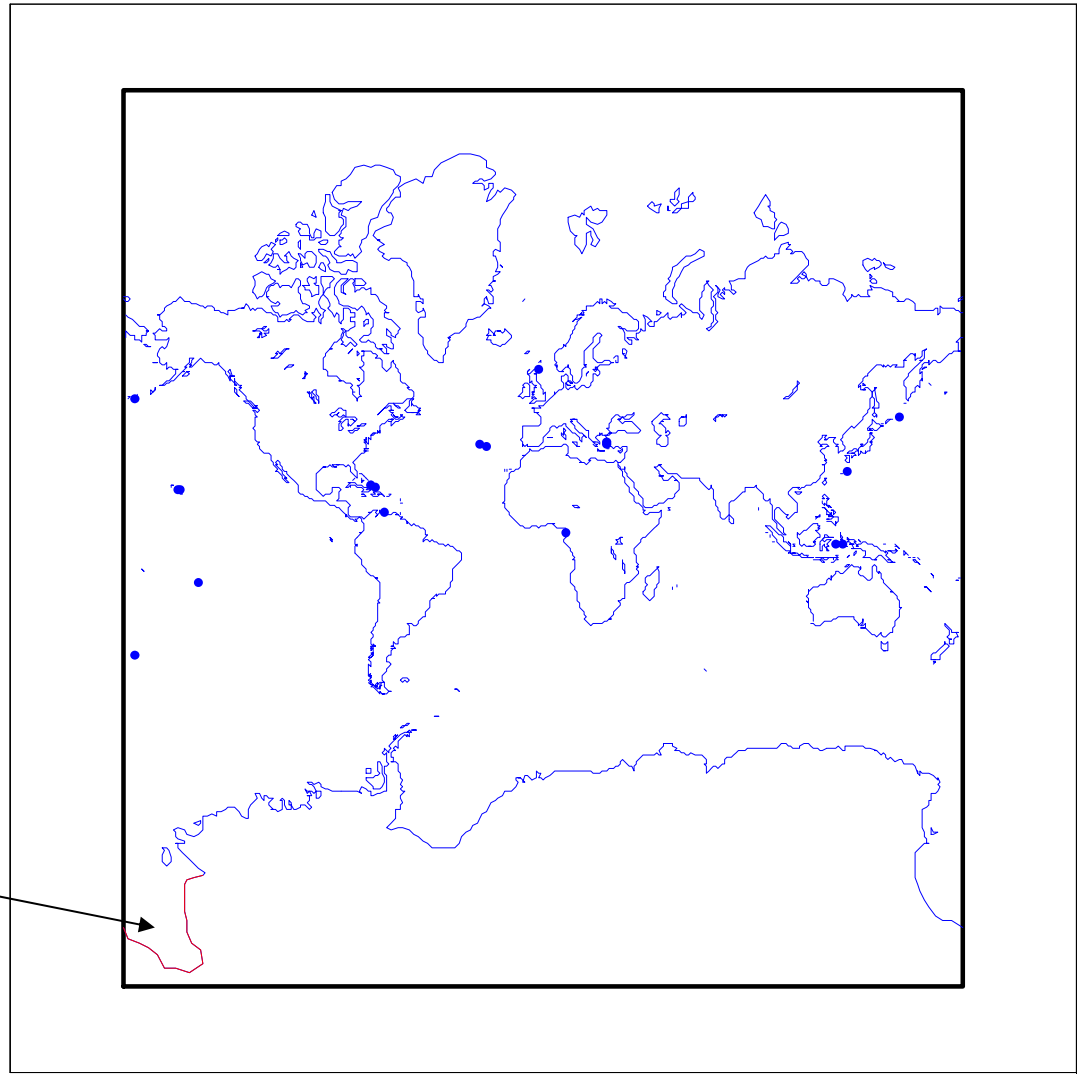
East longitudes are positive, whereas West longitudes are negative !!!!
Nord latitudes are positive, whereas Sud latitudes are negative!!!

Also note that a decimal degree notation is used, so that a longitude of 120 30'W is specified as -120.5. .

MATLAB mapping toolbox

Plotting vector geodata

```
>>clear
>> load coast
>> axesm mercator
>> framem
>> plotm(lat,long)
whos
   lat    9589x1      76712  double
  long    9589x1      76712  double

plotm(lat(1:20),long(1:20),'r')
```

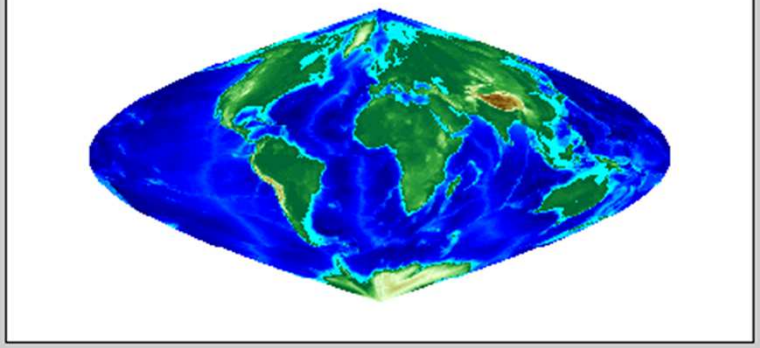


Plotting raster (grid) data in MATLAB

```
>> load topo
>> whos
  Name            Size              Bytes  Class     Attributes

  topo          180x360            518400  double
  topolegend      1x3                  24  double
  topomap1       64x3                1536  double
  topomap2      128x3                3072  double
>> axesm sinusoid
>> geoshow(topo,topolegend,'DisplayType','texturemap')
>> demcmap(topo)
>> topolegend
>> topolegend
topolegend =
     1    90     0
```

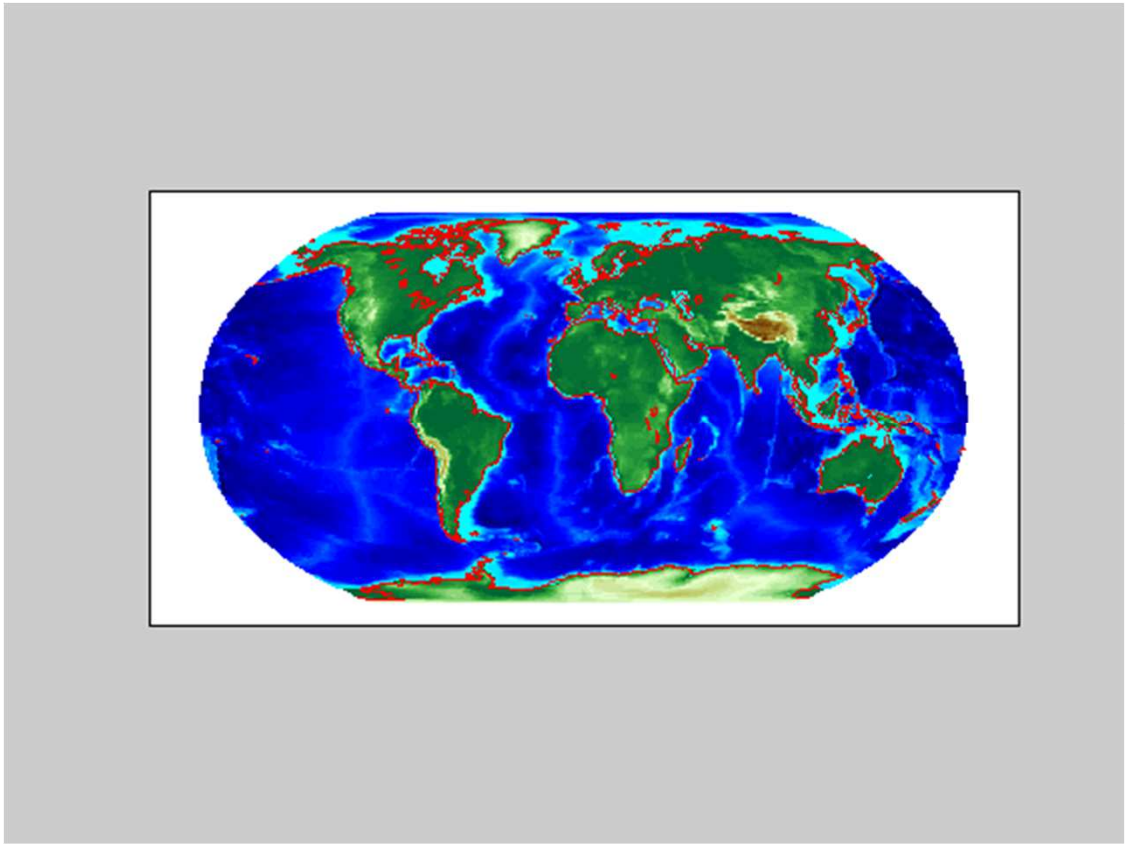
geoshow shows geodata in geographic unprojected ccordinates

demcmap creates and assigns a colormap appropriate for elevation data.



Combining vector and raster data in the same plot

```
>> clma
>> clear
>> load coast
>> load topo
>> axesm robinson
>> geoshow(topo,topolegend,'Displaytype','texturemap')
>> demcmap(topo)
>> geoshow(lat,long,'Color','r')
```



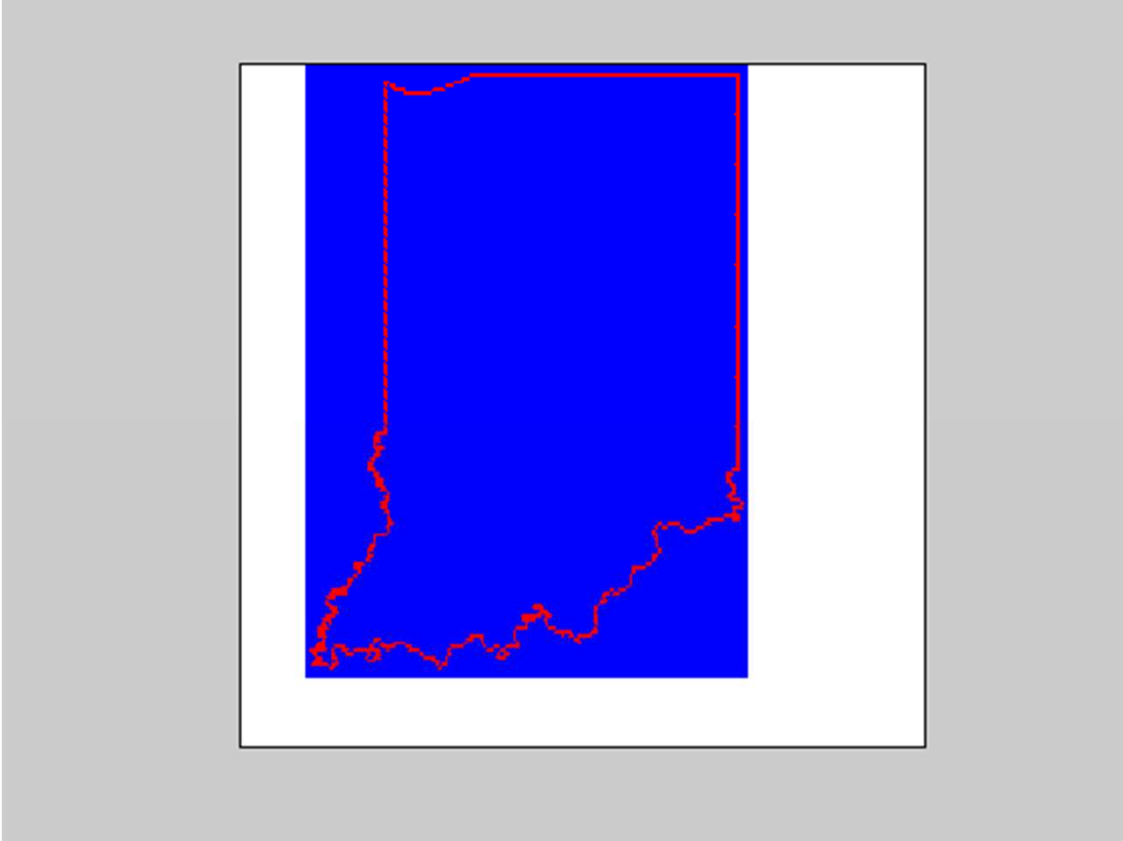
From vectors to raster (grids)

```
>> indiana=shaperead('usastatehi.shp','UseGeoCoords',true,'Selector',...
{@(name)strcmpi('Indiana',name),'Name'});
>> inLat=indiana.Lat;
>> inLon=indiana.Lon;
>> gridDensity=40;
>> [inGrid,inRefVec]=vec2mtx(inLat,inLon,gridDensity);
>> whos
  Name            Size              Bytes  Class     Attributes

  gridDensity     1x1                   8  double
  inGrid        164x137            179744 double
  inLat           1x626              5008 double
  inLon           1x626              5008 double
  inRefVec        1x3                  24 double
  indiana         1x1                10960 struct
```

Convert vector information to a raster grid image

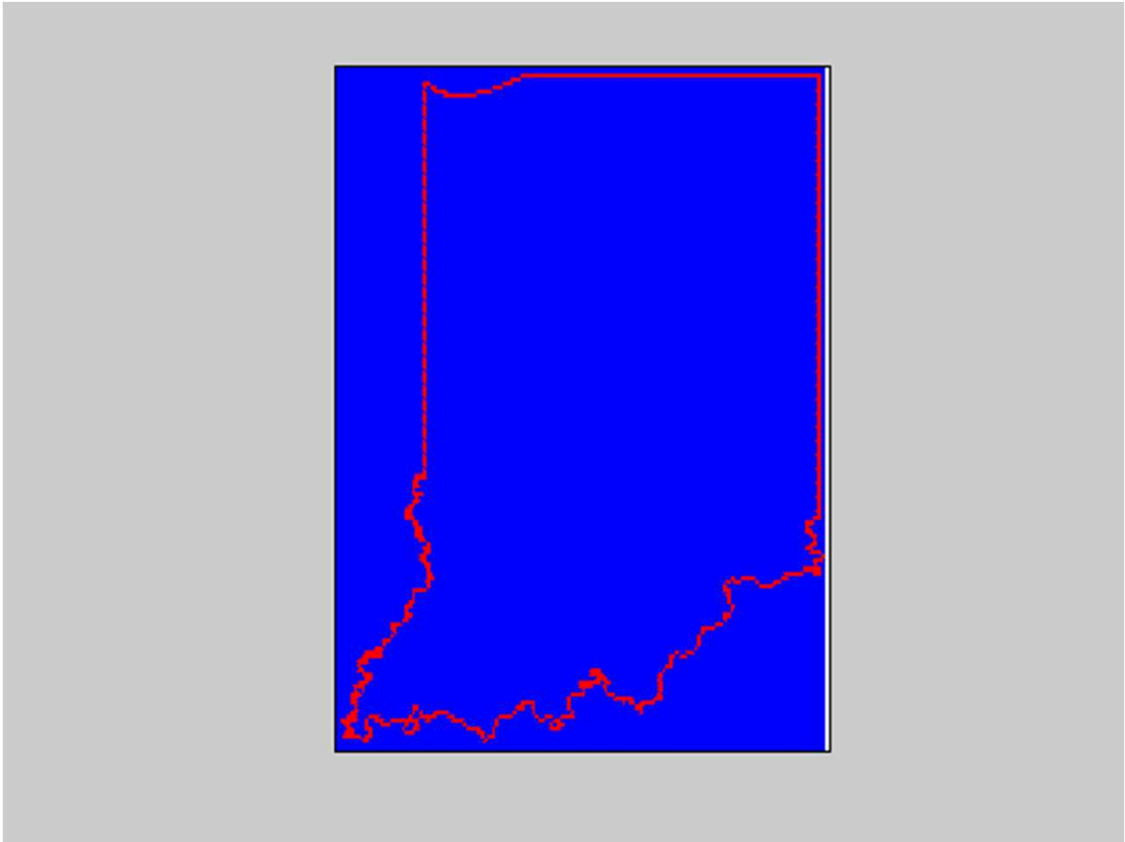
From vectors to rastrer (grids)

```
>> figure
>> axesm eqdcyl
>> meshm(inGrid,inRefVec)
>> colormap jet(4)
```



From vectors to rastrer (grids)

```
>> [latlim,lonlim]=limitm(inGrid,inRefVec)
latlim =
   37.7107   41.8107
lonlim =
  -88.1479  -84.7229
>> setm(gca,'Flatlimit',latlim,'Flonlimit',lonlim)
>> tightmap
```



Mapping

Mapping

Introduction to MATLAB mapping Toolbox

Map projections

MATLAB mapping Toolbox and georeferencing

Laboratory exercises:

Romà Tauler (IDAEA, CSIC, Barcelona)

Map Projections

All geospatial data must be flattened onto a display surface in order to visually portray what exists where. The mathematics and craft of map projection are central to this process. Although there is no limit to the ways geodata can be projected, conventions, constraints, standards, and applications generally prescribe its usage.

To represent a curved surface such as the Earth in two dimensions, you must geometrically transform (literally, and in the mathematical sense, “map”) that surface to a plane. Such a transformation is called a map projection.

map projection consists of constructing points on geometric objects such as cylinders, cones, and circles that correspond to homologous points on the surface of the planet being mapped according to certain rules and formulas.

Map Projections

But what projection should I use?

This depends really on how large an area you are mapping.

Usually, maps of the whole world are Mercator, although often the Miller Cylindrical projection looks better because it doesn't emphasize the polar areas as much.

Another choice is the Hammer-Aitoff or Mollweide (which has meridians curving together near the poles). Both are equal-area. It's probably not a good idea to use these projections for maps that don't have the equator somewhere near the middle.

The Robinson projection is not equal-area or conformal, but was the choice of National Geographic (for a while, anyway), and also appears in the IPCC reports.

If you are plotting something with a large north/south extent, but not very wide (say, North and South America, or the North and South Atlantic), then the Sinusoidal or Mollweide projections will look pretty good.

Another choice is the Transverse Mercator, although that is usually used only for very large-scale maps.

Map Projections
MATLAB mapping toolbox

Projection Name	Map Projection ID
Transverse Mercator	tranmerc
Mercator	mercator
Lambert Conformal Conic (2SP)	lambert
Lambert Conformal Conic (1SP)	lambert
Lambert Azimuthal Equal Area	eqaazim
Albers Equal-Area Conic	eqaconic
Azimuthal Equidistant	eqdazim
Equidistant Conic	eqdconic
Polar Stereographic	ups
Oblique Stereographic	stereo
Equirectangular	eqdcylin
Cassini-Soldner	cassini
Gnomonic	gnomonic
Miller Cylindrical	miller
Orthographic	ortho
Polyconic	polycon
Robinson	robinson
Sinusoidal	sinusoid
Van der Grinten	vgrint1

Map Projections

But what projection should I use?

For smaller areas within one hemisphere or other (say, Australia, the United States, the Mediterranean, the North Atlantic) you might pick a conic projection. The differences between the two available conic projections are subtle, and if you don't know much about projections it probably won't make much difference which one you use.

If you get smaller than that, it doesn't matter a whole lot which projection you use. One projection useful in many cases is the Oblique Mercator, since you can align it along a long (but narrow) coastal area.

If map limits along lines of longitude/latitude are OK, use a Transverse Mercator or Conic Projection. The UTM projection is also useful.

Polar areas are traditionally mapped using a Stereographic projection, since for some reason it looks nice to have a "bullseye" pattern of latitude lines.
If you want to get a quick idea of what any projection looks like, default parameters for all functions are set for a "typical" usage, i.e. to get a quick idea of what any projection looks like, you can do so without having to figure out a lot of numerical values:

How to specify map projections in MATLAB mapping Toolbox?

axesm Define map axes and set map properties

displaym Display geographic data from display structure

geoshow Display map latitude and longitude data

grid2image Display regular data grid as image

mapview Interactive map viewer

usamap Construct map axes for United States of America

worldmap Construct map axes for given region of world

MAPPING GUIs

AXESM GUI

AXESM activates a GUI to define a map projection for the current axes.
AXESM(PROPERTYNAME, PROPERTYVALUE,...)
AXESM(MSTRUCT,...)
AXESM(PROJFCN,...)

MAPVIEW Interactive map viewer

Use the Map Viewer to work with vector, image, and raster data grids in a map coordinate system: load data, pan and zoom on the map, control the map scale of your screen display, control the order, visibility, and symbolization of map layers, annotate your map, and click to learn more about individual vector features.

MAPVIEW complements MAPSHOW and GEOSHOW, which are for constructing maps in ordinary figure windows in a less interactive, script-oriented way.

MAPVIEW (with no arguments) starts a new Map Viewer in an empty state.

MAPSHOW Display map data without projection

MAKESYMBOLSPEC Construct vector symbolization specification

HELP for Mapping MATLAB Toolbox

- help map for computational functions
- mapdemos for a list of Mapping Toolbox demos
- maps lists all Mapping Toolbox map projections by class, name, and ID string.
- maplist returns a structure describing all Mapping Toolbox map projections.
- projlist to list map projections supported by projfwd and projinv
- help *functionname* for help on a specific function, often including examples
- helpwin *functioname* to see the output of help displayed in the Help browser window instead of the Command Window
- doc *functionname* to read a function's reference page in the Help browser, including examples and illustrations

MAPPING DEMOS

- mapexkmlexport — Exporting Vector Point Data to KML
- mapexfindcity — Interactive Global City Finder
- mapexgeo — Creating Maps Using geoshow (for latitude, longitude data)
- mapexmap — Creating Maps Using mapshow (for x, y data)
- mapexrefmat — Creating and Using Referencing Matrices
- mapexreg — Georeferencing an Image to an Orthotile Base Layer
- mapex3ddome — Plotting a 3-D Dome as a Mesh Over a Globe
- mapexunprojectdem — Un-Projecting a Digital Elevation Model (DEM)
- mapexgshhs — Converting Coastline Data (GSHHS) to Shapefile Format

MATLAB Functions that Access Internet Data Directly

The MATLAB functions in the table below read files from URLs. Then you can write the files to a local directory or load them into your MATLAB Workspace. With the exception of urlread and urlwrite, all the functions can read files on local and network file systems using path syntax as well.

geotiffread Read a georeferenced image from GeoTIFF file

gunzip Uncompress files in the GNU-Zip format

imread Read image from graphics file

untar Extract the contents of a Tar-file

unzip Extract the contents of a Zip-file

urlread Returns the contents of a URL as a string

urlwriteSave the contents of a URL to a file

Mapping

Mapping

Introduction to MATLAB mapping Toolbox

Data Models

Map projections

MATLAB mapping Toolbox and georeferencing

Laboratory exercises:

Romà Tauler (IDAEA, CSIC, Barcelona)

Georeferencing

Raster geodata consists of georeferenced data grids and images that in MATLAB are stored as matrices. Raster geodata looks like any other matrix in MATLAB

Georeferencing can be done to the earth globe or to a specified map projection, so that each pixel of data occupies a known patch of territory of the planet. Whether a raster geodata set covers the entire planet or not, its placement and resolution must be specified.

Referencing vectors

In many instances (when the data grid or image is based on latitude and longitude and is aligned with the geographic graticule), a referencing matrix has more degrees of freedom than the data requires. In such cases, you can use a more compact representation, a three-element *referencing vector*.

A referencing vector defines the pixel size and northwest origin for a regular, rectangular data grid:

refvec= [cells-per-degree, north lat, west long]

refvec= [cells-per-degree, north lat, west long]

In MAT-files, this variable is often called refvec or maplegend. The first element, cells-per-degree, describes the angular extent of each grid cell (e.g., if each cell covers five degrees of latitude and longitude, cells-per-degree would be specified as 0.2).

Note that if the latitude extent of cells differs from their longitude extent you cannot use a referencing vector, and instead must specify a referencing matrix.

The second element, north-lat, specifies the northern limit of the data grid (as a latitude), and the third element, west-lon, specifies the western extent of the data grid (as a longitude). In other words, north-lat, west-lon is the northwest corner of the data grid.

***Note, however, that cell (1,1) is always in the southwest corner of the grid*.** This need not be the case for grids or images described by referencing matrices, as opposed to referencing vectors.

Regular data grids

Dsta matrices that contain double values.

southermost edge is the first row
northermost edge is the last row
westernmost edge is the first column
eastermnost edge is the last column

cell(1,1) is southwest corner of the grid

East longitudes are positive, whereas West longitudes are negative !!!!

Nord latitudes are positive, whereas Sud latitudes are negative!!!

Building a global data grid

Build a coarse world map where each cell represents 60o

1) Built a data minigrid:

minigrid=[1,2,3,4,5,6;7,8,9,10,11,12;13,14,15,16,17,18]

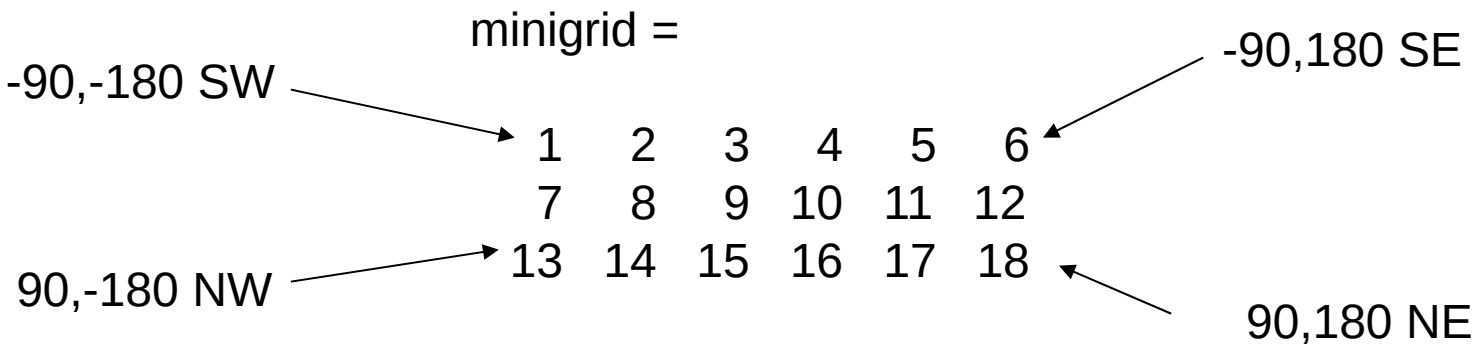
2) Make a referencing vector refvec= [cells-per-degree, north lat, west long]

minivec=[1/60,90,-180]

Building and georeferencing a global data grid

minigrid=[1,2,3,4,5,6;7,8,9,10,11,12;13,14,15,16,17,18]

minivec=[1/60,90,-180]



Remember!

East longitudes are positive.
West longitudes are negative.
North latitudes are positive.
South latitudes are negative.

3) Set up an equidistant cylindrical map projection

axesm('MapProjection','eqdcylin')

setm(gca,'GlineStyle','-','Grid','on','Frame','on')

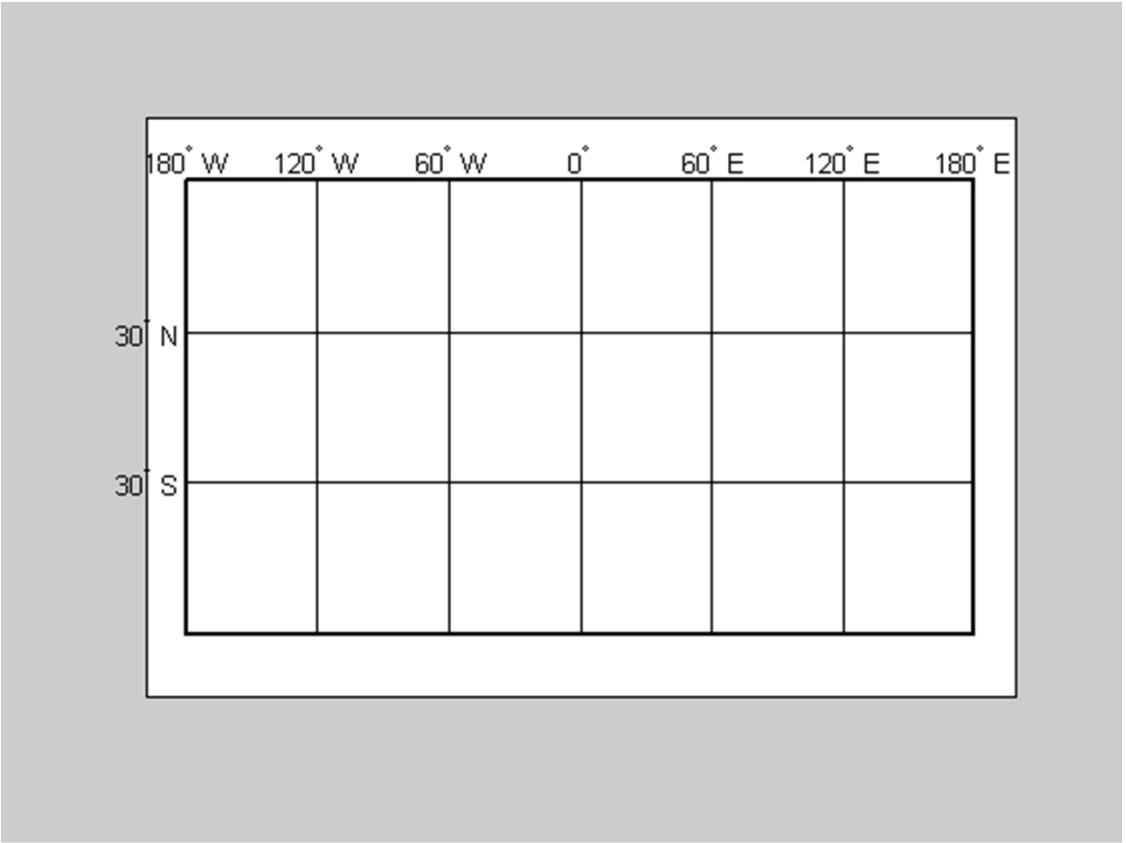
4) Draw a graticule with parallel and meridian labels at 60o intervals

setm(gca, 'MlabelLocation', 60,'PlabelLocation',[-30,30],...

'MlabelParallel','north','MeridianLabel','on',...

'ParallelLabel','on','MlineLocation',60,...

'PlineLocation',[-30,30])

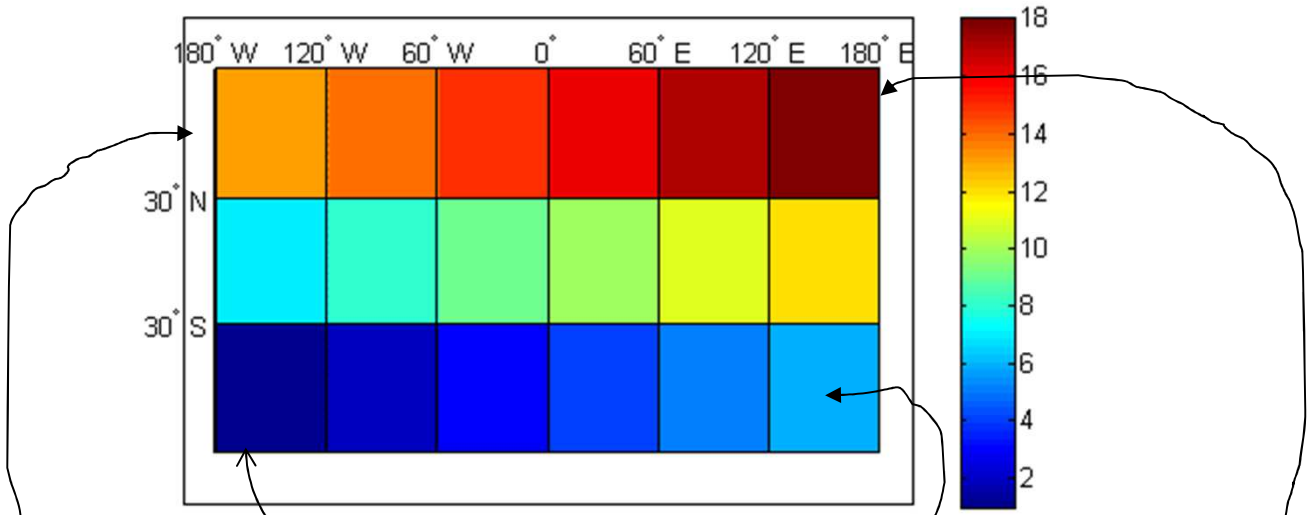


5) Map de data using mesm and display with a color map legend

minigrid=[1,2,3,4,5,6;7,8,9,10,11,12;13,14,15,16,17,18]

minivec=[1/60,90,-180]

```
>> meshm(minigrid,minivec);
>> colorbar
```



SW

SE

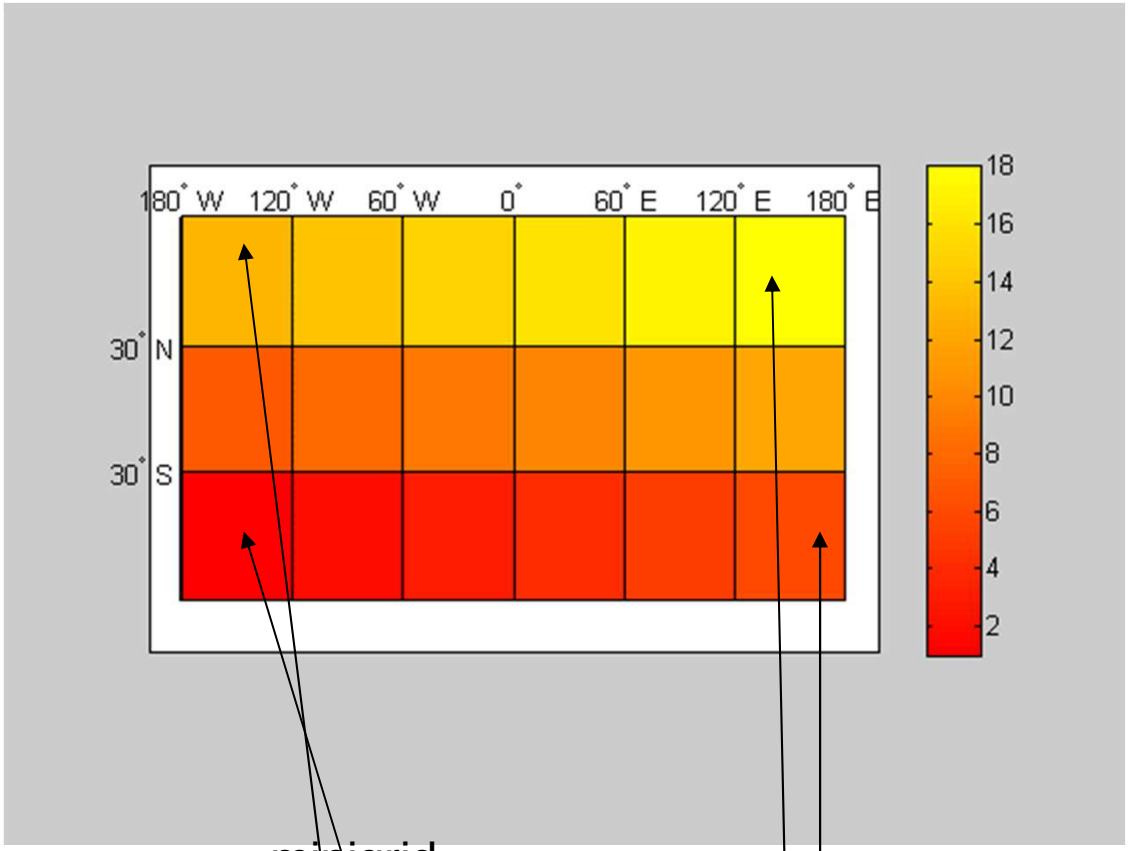
```
minigrid =
     1     2     3     4     5     6
     7     8     9    10    11    12
    13    14    15    16    17    18
```

NE

NW

5) Map de data using mesm and display with a color map legend

meshm(minigrid,minivec);
colormap('autumn');colorbar



minigrid =

-90,-180 SW	1	2	3	4	5	6	-90,180 SE
	7	8	9	10	11	12	
90,-180 NW	13	14	15	16	17	18	90,180 NE

Computing Map limits from reference vectors

```
load korea
refvec =   12.0000   45.0000  115.0000
[latlimits,longlimits]=limitm(map,refvec)
latlimits =
    30    45
longlimits =
   115   135
```

This means that korea region extends from 30^{o}N to 45^{o}N and 115^{o}E to 135^{o}E

```
>> [rows,cols]=size(map)
rows =
   180
cols =
   240
>> southlat=refvec(2)-rows/refvec(1)
southlat =
   30.0000
>> eastlon=refvec(3)+cols/refvec(1)
eastlon =
  135.0000
```

latitudes decrease southwards, longitudes increase eastwards

Geographic interpretation of matrix elements

```
>> load russia
>> [latlim,longlim]=limitm(map,refvec)
latlim =
    35    80
longlim =
    15   190
>> row=23;col=79;
[lat,long]=setltln(map,refvec,row,col)
lat =
   39.5000
long =
   30.7000
```

This means that the element row,column = 23,79 has this lat and long

```
>> [r,c]=setpostn(map,refvec,lat,long)
r =
    23
c =
    79
```

This means that this lat,long is in the 23,79 row coliumn

BUILDING A GEOREFERENCE MATRIX, R

Raster geodata is georeferenced in the toolbox through a companion data structure called a referencing matrix. This 3-by-2 matrix of doubles describes the scaling, orientation, and placement of the data grid on the globe. For a given referencing matrix, R, one of the following relations holds between rows and columns and coordinates (depending on whether the grid is based on map coordinates or geographic coordinates, respectively):

[x y] = [row col 1] * R or [long lat] = [row col 1] * R

MAKEREFMAT Construct affine spatial-referencing matrix
A spatial referencing matrix R ties the row and column subscripts of an image or regular data grid to 2-D map coordinates or to geographic coordinates (longitude and geodetic latitude). R is a 3-by-2 affine transformation matrix. R either transforms pixel subscripts (row, column) to/from map coordinates (x,y) according to

[x y] = [row col 1] * R

or transforms pixel subscripts to/from geographic coordinates according to

[lon lat] = [row col 1] * R.

BUILDING A GEOREFERENCE MATRIX, R

To construct a referencing matrix for use with geographic coordinates, use longitude in place of X and latitude in place of Y, as shown in the third syntax below. This is one of the few places where longitude precedes latitude in a function call.

R = MAKEREFMAT(X11, Y11, DX, DY) with scalar DX and DY constructs a referencing matrix that aligns image/data grid rows to map X and columns to map Y. X11 and Y11 are scalars that specify the map location of the center of the first (1,1) pixel in the image or first element of the data grid, so that

[X11 Y11] = pix2map(R,1,1).

DX is the difference in X (or longitude) between pixels in successive columns and DY is the difference in Y (or latitude) between pixels in successive rows. More abstractly, R is defined such that

[X11 + (col-1) * DX, Y11 + (row-1) * DY] = pix2map(R, row, col).

Mapping (MATLAB) georeferencing: makerefmat

Construct spatial referencing matrix of raster/grid files using command makerefmat.m

Example 1

Create a referencing matrix for an image with square, four-meter pixels and with its upper left corner (in a map coordinate system) at x = 207000 meters, y = 913000 meters. The image follows the typical orientation: x increasing from column to column and y decreasing from row to row.

```
x11 = 207002;  % Two meters east of the upper left corner
y11 = 912998;  % Two meters south of the upper left corner
dx =  4;
dy = -4;
R = makerefmat(x11, y11, dx, dy)
x11 = 207002;  % Two meters east of the upper left corner
y11 = 912998;  % Two meters south of the upper left corner
dx =  4;
dy = -4;
R = makerefmat(x11, y11, dx, dy)
R =
         0         -4
         4          0
    206998     913002
```

Example 2

Create a referencing matrix for a global geoid grid.

```
load geoid  % Adds array 'geoid' to the workspace

% 'geoid' contains a model of the Earth's geoid sampled in
% one-degree-by-one-degree cells.  Each column of 'geoid' contains
% geoid heights in meters for 180 cells starting at latitude
% -90 degrees and extending to +90 degrees, for a given latitude.
% Each row contains geoid heights for 360 cells starting at
% longitude 0 and extending 360 degrees.

lat11 = -89.5;  % Cell-center latitude corresponding to geoid(1,1)
lon11 =   0.5;  % Cell-center longitude corresponding to geoid(1,1)
dLat = 1;  % From row to row moving north by one degree
dLon = 1;  % From column to column moving east by one degree

geoidR = makerefmat(lon11, lat11, dLon, dLat)
```

```
% It's well known that at its most extreme the geoid reaches a
      % minimum of slightly less than -100 meters, and that the minimum
      % occurs in the Indian Ocean at approximately 4.5 degrees latitude,
      % 78.5 degrees longitude.  Check the geoid height at this location
      % by using LATLON2PIX with the new referencing matrix:

      [row, col] = latlon2pix(geoidR, 4.5, 78.5)
      geoid(round(row),round(col))

geoidR =
         0    1.0000
    1.0000         0
   -0.5000  -90.5000
row =
    95
col =
    79
ans =
 -106.9260
```

Example 3

Create a half-resolution version of a georeferenced TIFF image, using Image Processing Toolbox functions IND2GRAY and IMRESIZE.

```
% Read the indexed-color TIFF image and convert it to grayscale.
% The size of the image is 2000-by-2000.
[X, cmap] = imread('concord_ortho_w.tif');
I_orig = ind2gray(X, cmap);

% Read the corresponding worldfile.  Each image pixel covers a
% one-meter square on the map.
R_orig = worldfileread('concord_ortho_w.tfw')

% Halve the resolution, creating a smaller (1000-by-1000) image.
I_half = imresize(I_orig, size(I_orig)/2, 'bicubic');

% Find the map coordinates of the center of pixel (1,1) in the
% resized image: halfway between the centers of pixels (1,1) and
% (2,2) in the original image.
[x11_orig, y11_orig] = pix2map(R_orig, 1, 1)
[x22_orig, y22_orig] = pix2map(R_orig, 2, 2)
```

```
% Average these to determine the center of pixel (1,1) in the new
    % image.
    x11_half = (x11_orig + x22_orig) / 2
    y11_half = (y11_orig + y22_orig) / 2
% Construct a referencing matrix for the new image, noting that its
    % pixels are each two meters square.
    R_half = makerefmat(x11_half, y11_half, 2, -2)
% Display each image in map coordinates.
    figure;
    subplot(2,1,1); h1 = mapshow(I_orig,R_orig); ax1 = get(h1,'Parent');
    subplot(2,1,2); h2 = mapshow(I_half,R_half); ax2 = get(h2,'Parent');
    set(ax1, 'XLim', [208000 208250], 'YLim', [911800 911950])
    set(ax2, 'XLim', [208000 208250], 'YLim', [911800 911950])
% Mark the same map location on top of each image.
    x = 208202.21;
    y = 911862.70;
    line(x, y, 'Parent', ax1, 'Marker', '+', 'MarkerEdgeColor', 'r');
    line(x, y, 'Parent', ax2, 'Marker', '+', 'MarkerEdgeColor', 'r');
% Graphically, they coincide, even though the same map location
    % corresponds to two different pixel coordinates.
    [row1, col1] = map2pix(R_orig, x, y)
    [row2, col2] = map2pix(R_half, x, y)
```

Mapping

Mapping

Introduction to MATLAB mapping Toolbox
Map projections
MATLAB mapping Toolbox and georeferencing

Laboratory exercises:

Plotting images and maps

Google map loader
Getting satellite data from internet
Representing results of a monitoring campaign

Romà Tauler (IDAEA, CSIC, Barcelona)

Reading images

IMREAD Read image from graphics file.
A = IMREAD(FILENAME,FMT) reads a grayscale or color image from the file specified by the string FILENAME. If the file is not in the current directory, or in a directory on the MATLAB path, specify the full pathname.
The text string FMT specifies the format of the file by its standard file extension. For example, specify 'gif' for Graphics Interchange Format files. To see a list of supported formats, with their file extensions, use the IMFORMATS function. If IMREAD cannot find a file named FILENAME, it looks for a file named FILENAME.FMT.
The return value A is an array containing the image data. If the file contains a grayscale image, A is an M-by-N array. If the file contains a truecolor image, A is an M-by-N-by-3 array. For TIFF files containing color images that use the CMYK color space, A is an M-by-N-by-4 array. See TIFF in the Format-Specific Information section for more information.

Look at C:\Archivos de programa\MATLAB\R2007a\toolbox\images\imdemos for images and demos

.....

imformats

EXT	ISA	INFO	READ	WRITE	ALPHA	DESCRIPTION
bmp	isbmp	imbmpinfo	readbmp	writebmp	0	Windows Bitmap (BMP)
cur	iscur	imcurinfo	readcur		1	Windows Cursor resources (CUR)
fts fits	isfits	imfitsinfo	readfits		0	Flexible Image Transport System (FITS)
gif	isgif	imgifinfo	readgif	writegif	0	Graphics Interchange Format (GIF)
hdf	ishdf	imhdfinfo	readhdf	writehdf	0	Hierarchical Data Format (HDF)
ico	isico	imicoinfo	readico		1	Windows Icon resources (ICO)
jpg jpeg	isjpg	imjpginfo	readjpg	writejpg	0	Joint Photographic Experts Group (JPEG)
pbm	ispbm	impnminfo	readpnm	writepnm	0	Portable Bitmap (PBM)
pcx	ispcx	impcxinfo	readpcx	writepcx	0	Windows Paintbrush (PCX)
pgm	ispgm	impnminfo	readpnm	writepnm	0	Portable Graymap (PGM)
png	ispng	impnginfo	readpng	writepng	1	Portable Network Graphics (PNG)
pnm	ispnm	impnminfo	readpnm	writepnm	0	Portable Any Map (PNM)
ppm	isppm	impnminfo	readpnm	writepnm	0	Portable Pixmap (PPM)
ras	isras	imrasinfo	readras	writeras	1	Sun Raster (RAS)
tif tiff	istif	imtifinfo	readtif	writetif	0	Tagged Image File Format (TIFF)
xwd	isxwd	imxwdinfo	readxwd	writexwd	0	X Window Dump (XWD)

```
A=imread('moon.tif')
image(A)
imagesc(A)
;
```

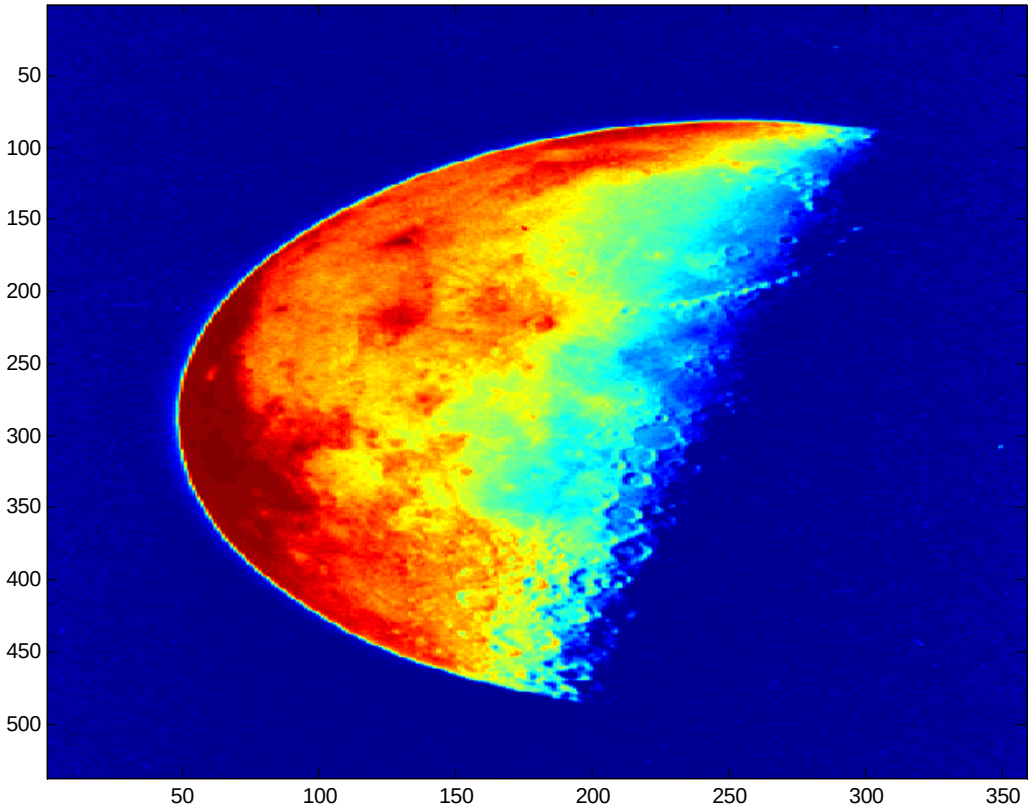
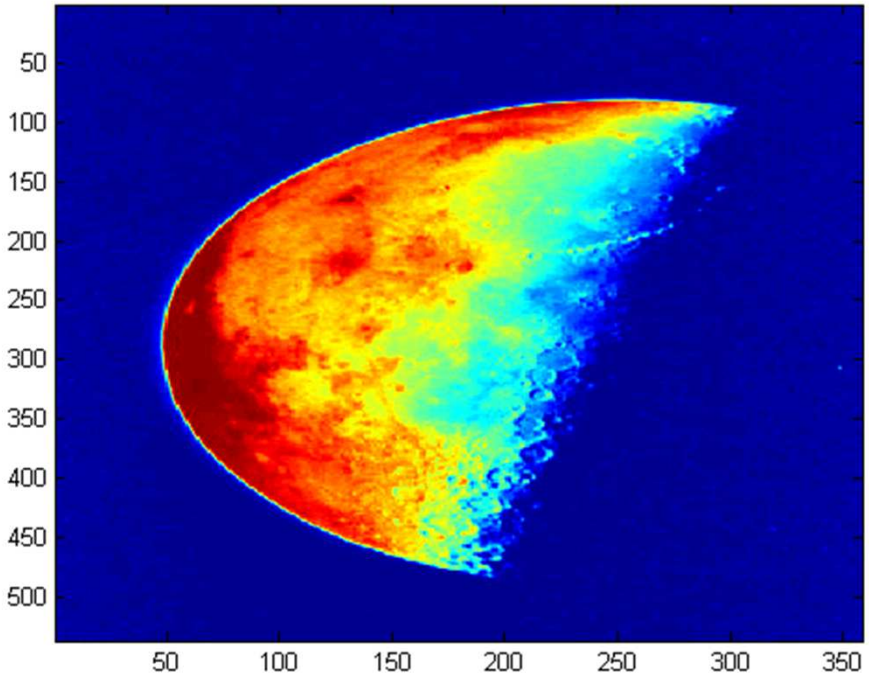


Table: summary of supported image types

BMP	1-bit, 8-bit and 24-bit uncompressed images
GIF	8-bit images
HDF	8-bit raster image datasets, with or without associated colormap; 24-bit raster image datasets; uncompressed or with RLE or JPEG compression
JPEG	8-bit, 12-bit, and 16-bit Baseline JPEG images
PBM	Any 1-bit PBM image, ASCII (plain) or raw (binary) encoding.
PCX	8-bit images
PGM	Any standard PGM image. ASCII (plain) encoded with arbitrary color depth. Raw (binary) encoded with up to 16 bits per gray value.
PNG	1-bit, 2-bit, 4-bit, 8-bit, and 16-bit grayscale images; 8-bit and 16-bit grayscale images with alpha channels; 1-bit, 2-bit, 4-bit, and 8-bit indexed images; 24-bit and 48-bit truecolor images; 24-bit and 48-bit truecolor images with alpha channels
PNM	Any of PPM/PGM/PBM (see above) chosen automatically
PPM	Any standard PPM image. ASCII (plain) encoded with arbitrary color depth. Raw (binary) encoded with up to 16 bits per color component.
RAS	Any RAS image, including 1-bit bitmap, 8-bit indexed, 24-bit truecolor and 32-bit truecolor with alpha
TIFF	Baseline TIFF images, including 1-bit, 8-bit, 16-bit, and 24-bit uncompressed images; 1-bit, 8-bit, 16-bit, and 24-bit images with packbits compression; 1-bit images with CCITT 1D, Group 3, and Group 4 compression; CIELAB, ICCLAB, and CMYK images
XWD	8-bit ZPixmaps

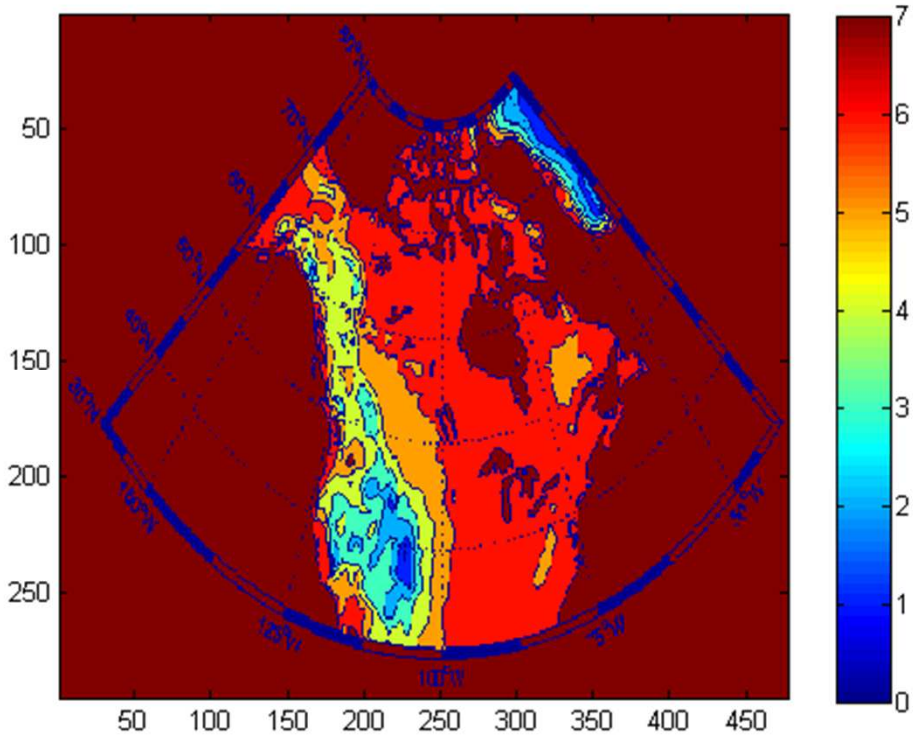
imwrite(x,filename,fmt)

```
>> A=imread('moon.tif');
>> imagesc(A)
>>
imwrite(A,'moon.jpeg','jpeg');
>> clear
>> A=imread('moon.jpeg');
>> imagesc(A)
```



...\m_map\private

```
A=imread('exlamber.gif');
image(A)
imagesc(A)
```



Mapping

Mapping

Introduction to MATLAB mapping Toolbox
Data Models
Map projections
MATLAB mapping Toolbox and georeferencing
Introduction to M_map Toolbox

Laboratory exercises:

Plotting images
Google maps and Google Earth
Getting satellite data from internet
Representing results of a monitoring campaign

Romà Tauler (IDAEA, CSIC, Barcelona)

LOADING A MAP AS AN IMAGE FROM GOOGLE MAPS

GEOREFERENCING IT AND PLOTTING VALUES

PLOTTING SYMBOLS ACCORDING THE SIZE OF A VARIABLE (E.G. FLUORESCENCE)

EXAMPLE

Google map of the river region was read as a ‘7.tif’ file
GPS limits of the map are in gpslimit (aprox. from Google)
GPS positions of sampling pointys are in gpsxy
Values to represent are in variable z
All these variables have been stored in mapex.mat

```
>> load('mapex.mat')
>> whos
  Name            Size            Bytes  Class     Attributes

  gpslimit        2x2                32  double
  gpsxy           62x2              992  double
  sampnumbers     1x62              496  double
  z               122x1             976  double
```

```
function mapdatagpsRT(gpslimit,gpsxy,sampnumbers,z)

% function mapdatagps(gpslimit);

% INPUT

% gpslimit=[lat1,long1;lat2,long2]

% lat1,long1 is right down corner

% lat2,long2 is left up corner

% example
gpslimit=[42.020003,2.426382;41.940318,2.217402];

% gpsxy are the GPS values where the representation
has to be performed

% gpsxy(lat,long)

% sampnumbers are sample numbers used for
representation
```

>> mapdatagpsRT(gpslimit,gpsxy,sampnumbers,z)

% scatter(y(i),x(i),(z(i)-min(z)+0.1)*100,[a,b,c],'.')

% scatter(y(i),x(i),(z(i)-min(z)+0.1)*20,'o','MarkerEdgeColor','k','MarkerFaceColor','c')

Test with different symbols, sizes and color….

Program plot_google_map

```
function varargout = plot_google_map(varargin)
% function h = plot_google_map(varargin)
% Plots a google map on the current axes using the Google Static Maps API
%
% USAGE:
% h = plot_google_map(Property, Value,...)
% Plots the map on the given axes. Used also if no output is specified
%
% Or:
% [lonVec latVec imag] = plot_google_map(Property, Value,...)
% Returns the map without plotting it
%
```

```
% PROPERTIES:
%    Height (640)   - Height of the image in pixels (max 640)
%    Width  (640)   - Width of the image in pixels (max 640)
%    Scale (2)      - (1/2) Resolution scale factor . using Scale=2 will
%                     double the resulotion of the downloaded image (up
%                     to 1280x1280) and will result in finer rendering,
%                     but processing time will be longer.
%    MapType        - ('roadmap') Type of map to return. Any of [roadmap,
%                     satellite, terrain, hybrid] See the Google Maps API for
%                     more information.
%    Alpha (1)      - (0-1) Transparency level of the map (0 is fully
%                     transparent). While the map is always
%                     moved to the bottom of the plot (i.e. will
%                     not hide previously drawn items), this can
%                     be useful in order to increase readability
%                     if many colors are ploted (using SCATTER
%                     for example).
```

```
%    Language       - (string) A 2 letter ISO 639-1 language code for displaying
labels in a local language instead of English (where available).
%                   For example, for Chinese use:
%                   plot_google_map('language','zh')
%                   For the list of codes, see:
%                   http://en.wikipedia.org/wiki/List_of_ISO_639-1_codes
%    Marker         - The marker argument is a text string with fields
%                   conforming to the Google Maps API. The
%                   following are valid examples:
%                   '43.0738740,-70.713993' (default midsize orange marker)
%                   '43.0738740,-70.713993,blue' (midsize blue marker)
%                   '43.0738740,-70.713993,yellowa' (midsize yellow
%                   marker with label "A")
%                   '43.0738740,-70.713993,tinyredb' (tiny red marker
%                   with label "B")
%    Refresh (1)    - (0/1) defines whether to automatically refresh the
%                   map upon zoom/pan action on the figure.
```

```
% OUTPUT:
%    h             - Handle to the plotted map
%    lonVect       - Vector of Longidute coordinates (WGS84) of the image
%    latVect       - Vector of Latidute coordinates (WGS84) of the image
%    imag          - Image matrix (height,width,3) of the map
% EXAMPLE - plot a map showing some capitals in Europe:
%    lat = [48.8708   51.5188   41.9260   40.4312   52.523   37.982];
%    lon = [2.4131    -0.1300    12.4951   -3.6788    13.415   23.715];
%    plot(lon,lat,'.r','MarkerSize',20)
%    plot_google_map
% References:
% http://www.mathworks.com/matlabcentral/fileexchange/24113
% http://www.maptiler.org/google-maps-coordinates-tile-bounds-projection/
% http://developers.google.com/maps/documentation/staticmaps/
%  Acknowledgement to Val Schmidt for his submission of
get_google_map.m
```

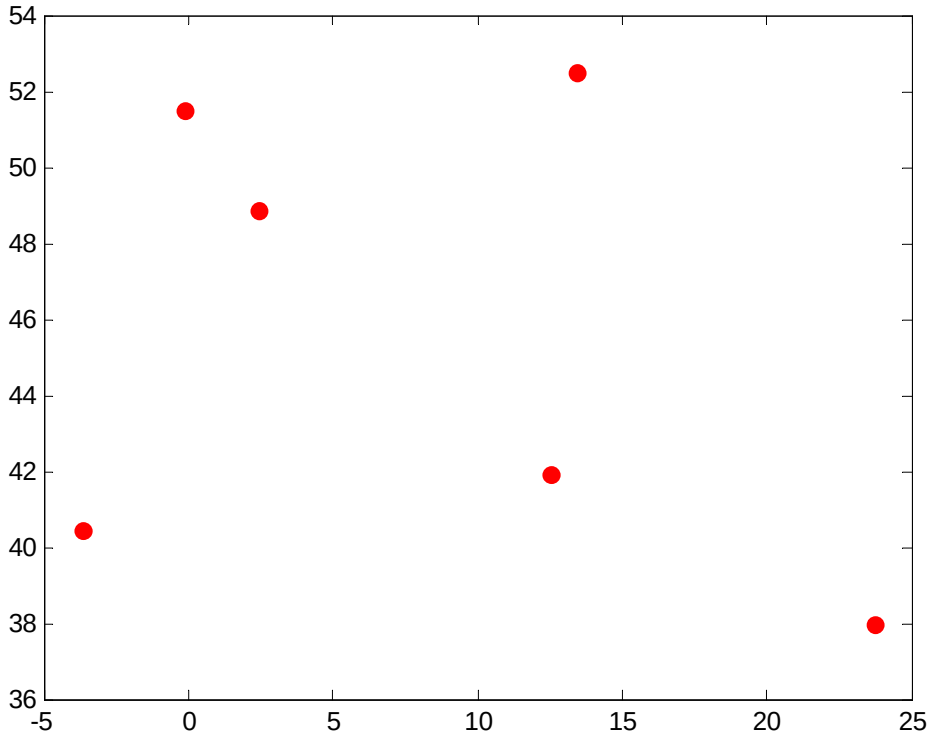
plot_google_map

% EXAMPLE - plot a map showing some capitals in Europe:

% lat = [48.8708 51.5188 41.9260 40.4312 52.523 37.982];

% lon = [2.4131 -0.1300 12.4951 -3.6788 13.415 23.715];

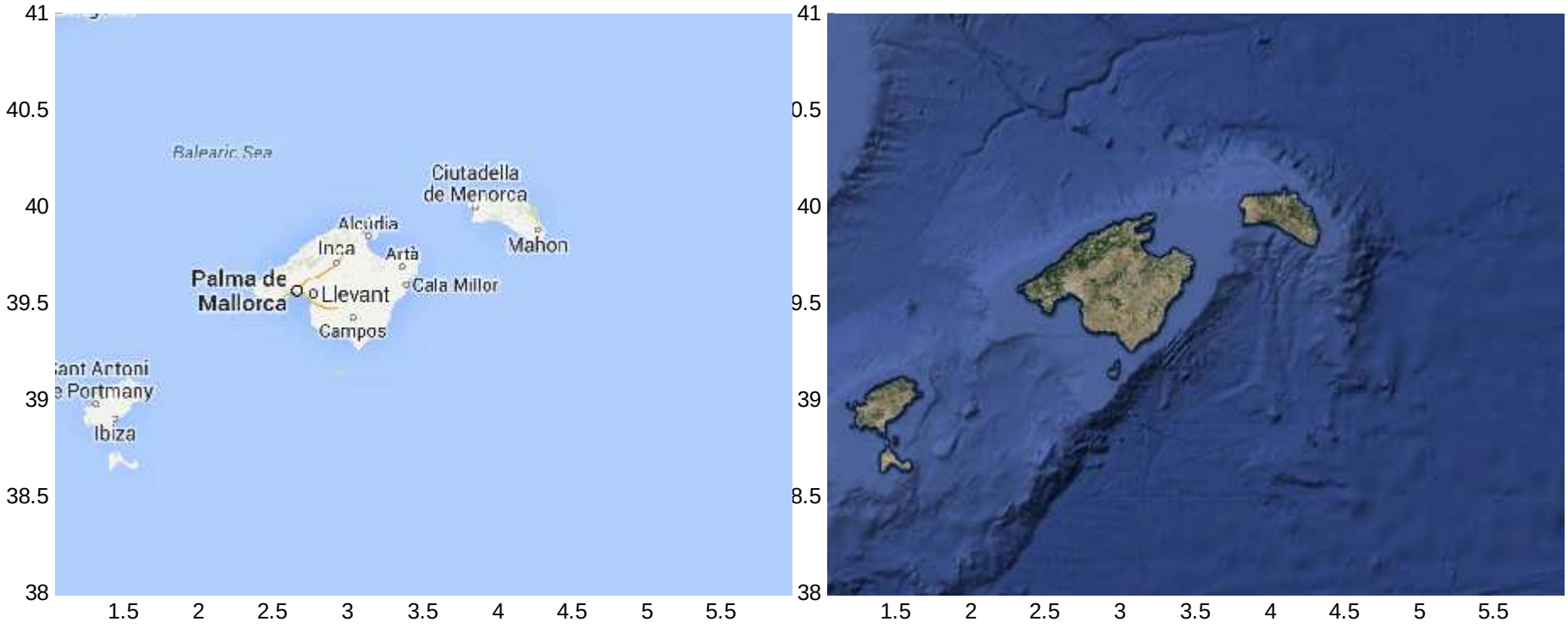
% plot(lon,lat,'.r','MarkerSize',20)



plot_google_map

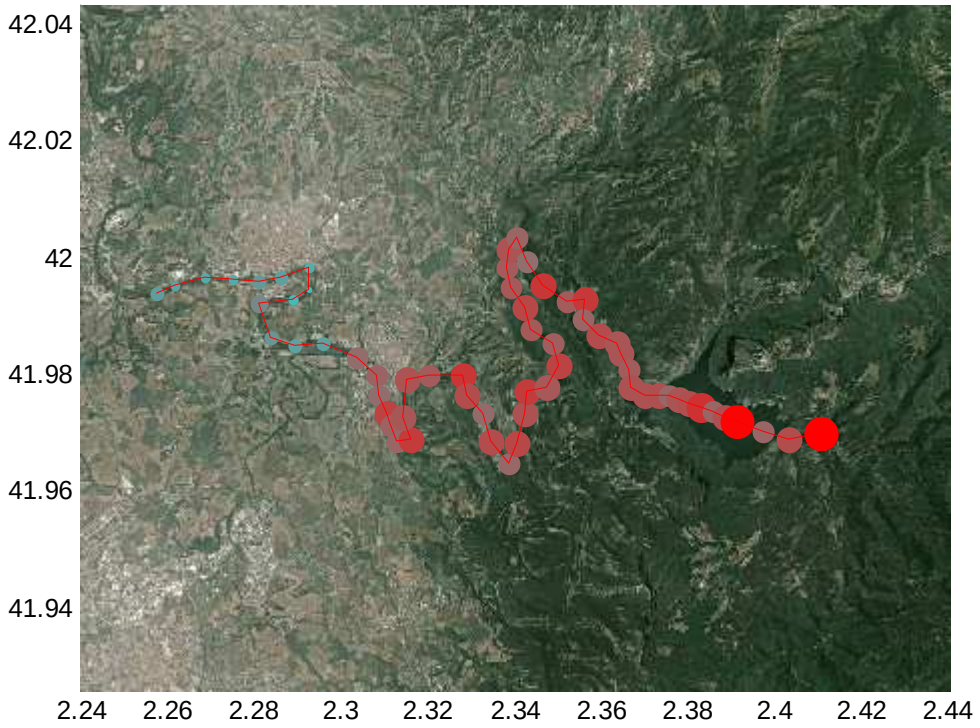
```
>> axis([0,5,40,44]);
>> plot_google_map
```

```
>> axis([2,5,38,41]);
>> plot_google_map('maptype','roadmap');
>> plot_google_map('maptype','satellite');
```



>> load mapex

>> mapgoogle(gpsxy,sampnumbers,z)



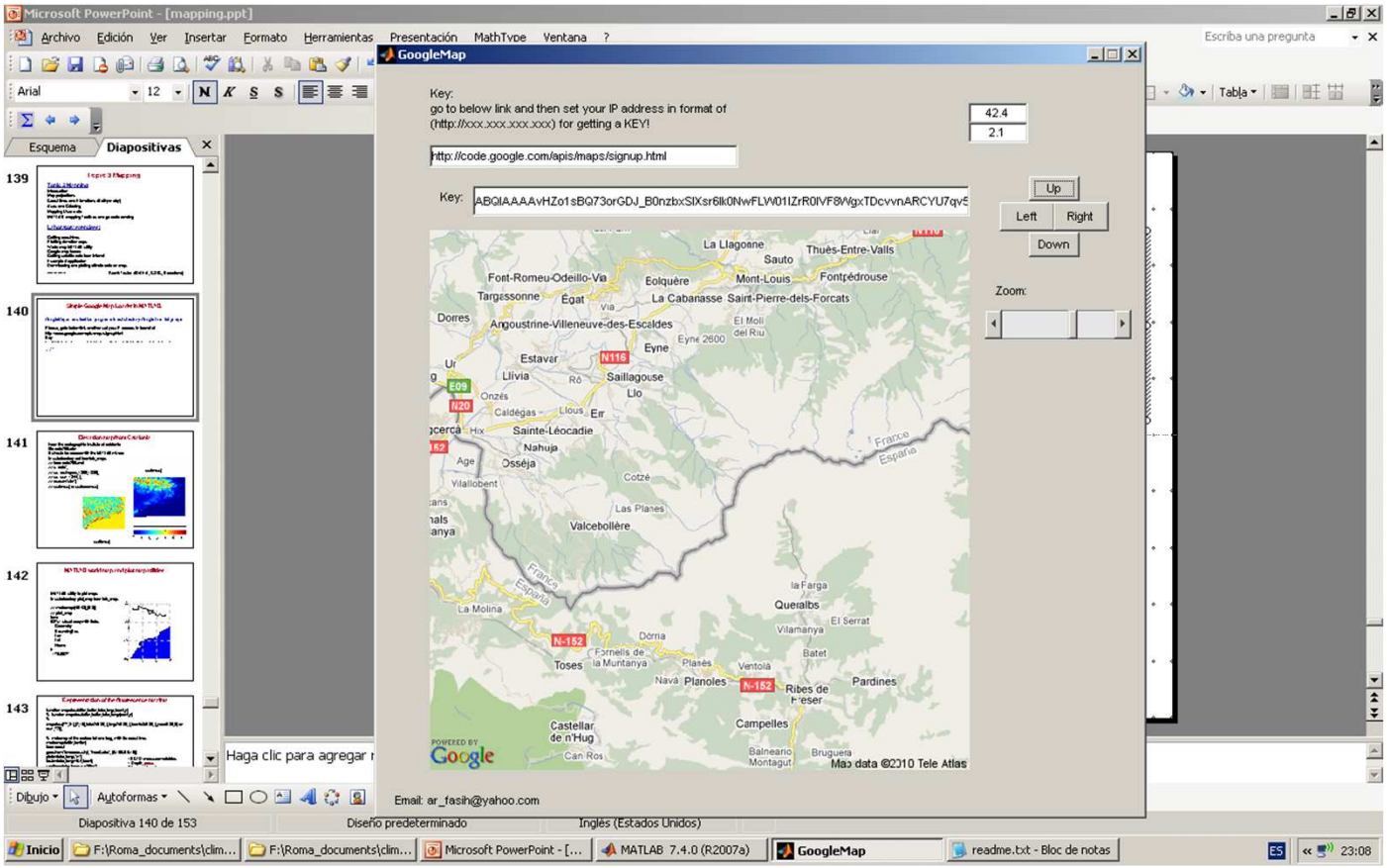
Other possibilities???

Simple Google Map Loader in MATLAB.

GoogleMap.m and test1.m programs in subdirectory Google from lab_maps

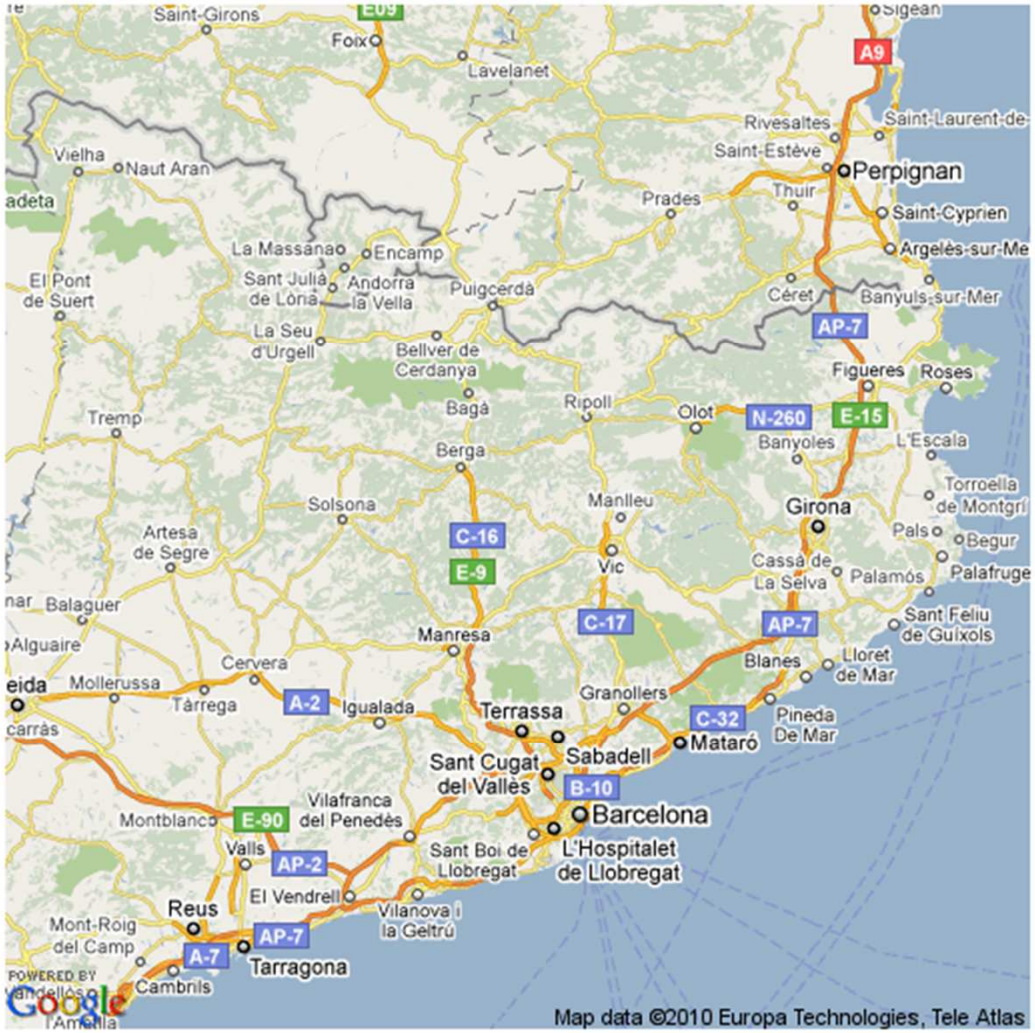
Please, go to below link and then set your IP address in format of
http://code.google.com/apis/maps/signup.html
Key:
ABQIAAAAvHZo1sBQ73orGDJ_B0nzbxSIXsr6Ik0NwFLW01IZrR0IVF8WgxTDcvvnARCYU7qv5P59bACbpikqdw

Nord 42
Est 2.0



Simple Google Map Loader in MATLAB.

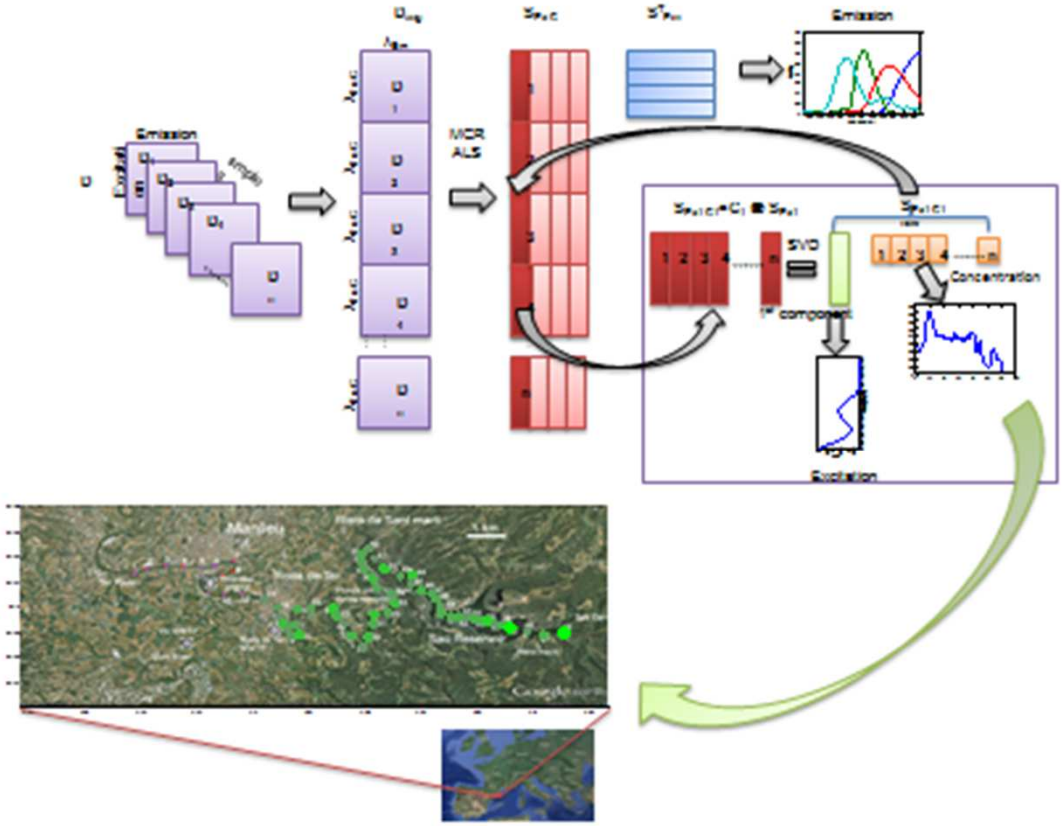
```
% Simple Google Map Loader in MATLAB.

% Key:

% Please, go to below link and then set your IP address in format of

% (http://xxx.xxx.xxx.xxx) for getting a KEY!

% http://code.google.com/apis/maps/signup.html

clc

clear all

key='ABQIAAAAteHND9YP6ctZEpCJsGZpWxRQWo9-sWXbVvdU2wQSDpuKctuXhBQYpf8Mm2875572Jwd2ge0XshBKBg';

pos='http://maps.google.com/staticmap?center=42.0,2.0&zoom=8&size=512x512';

address=strcat(pos,'&key=',key);

[I map]=imread(address,'gif');

RGB=ind2rgb(I,map);

imshow(RGB);
```



Distribution of dissolved organic matter in freshwaters using excitation emission fluorescence and Multivariate Curve Resolution

Xin Zhang, Rafael Marcé, Joan Armengol, Romà Tauler

Chemosphere 111 (2014) 120–128



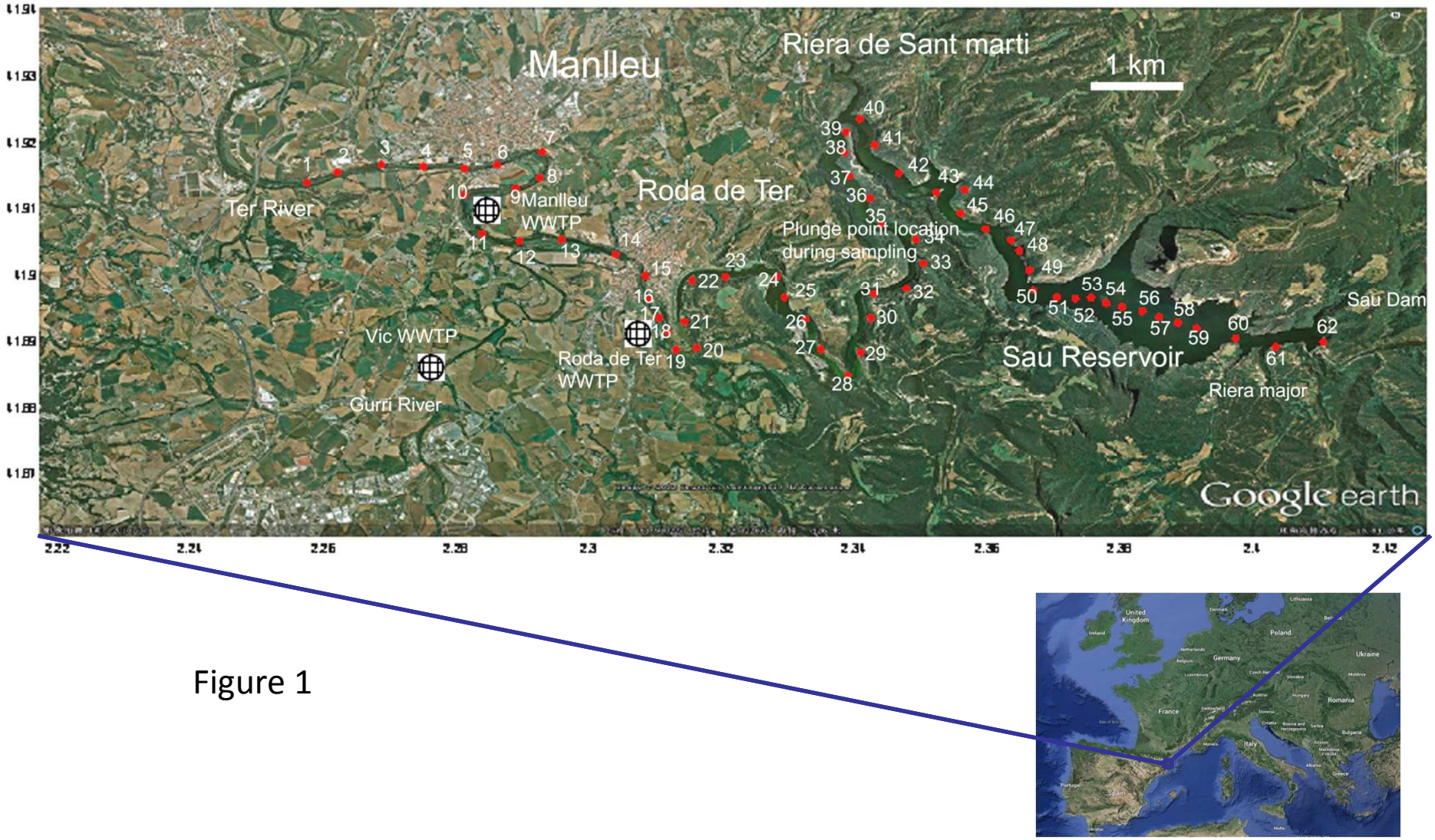

Manlleu
Riera de Sant marti
1 km
Ter River
Manlleu WWTP
Roda de Ter
Plunge point location during sampling
Vic WWTP
Gurri River
Roda de Ter WWTP
Sau Reservoir
Sau Dam
Riera major
Google earth
2.22
2.24
2.26
2.28
2.3
2.32
2.34
2.36
2.38
2.4
2.42

Figure 1

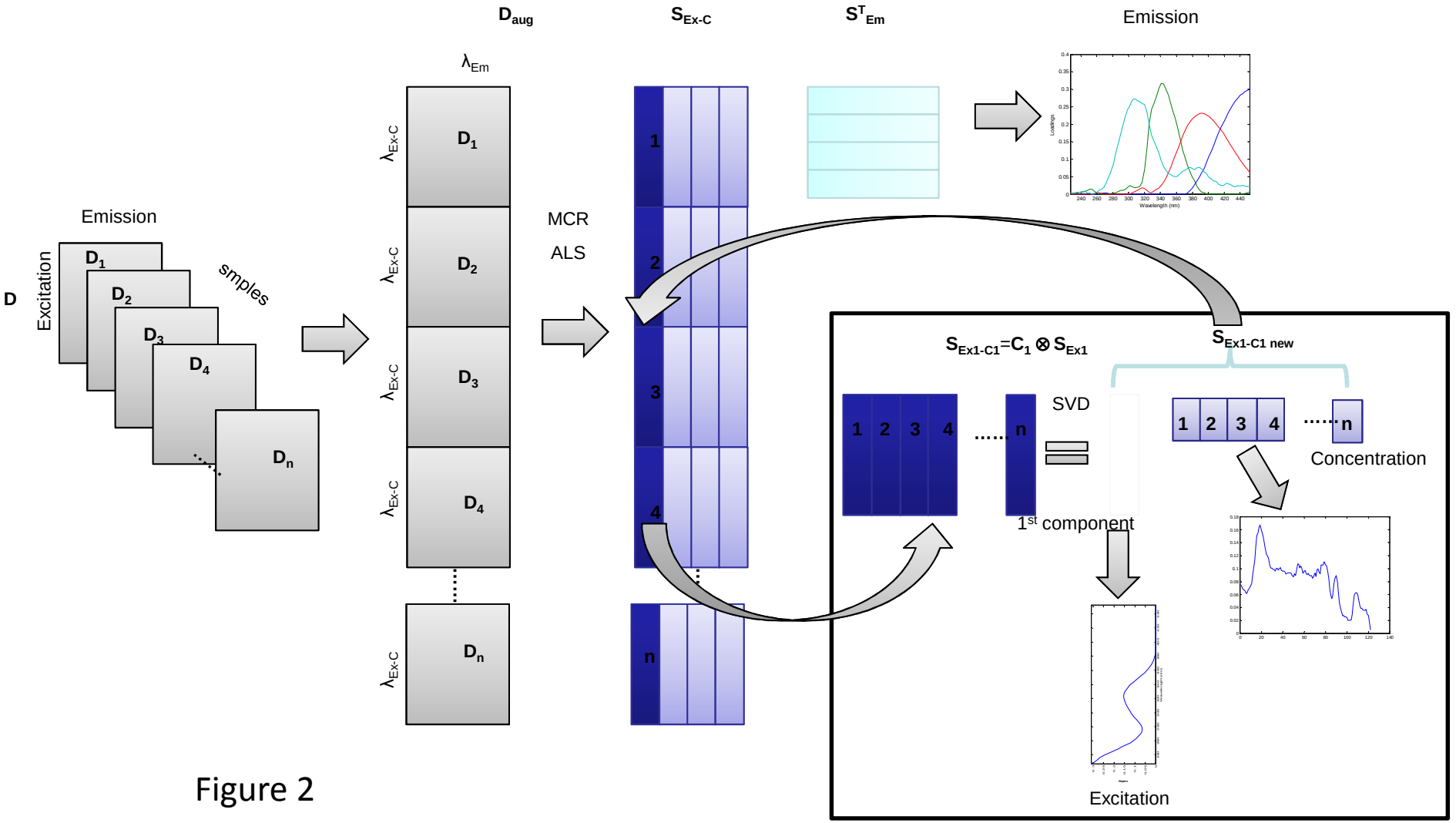

D
Excitation
Emission
D1
D2
D3
D4
Dn
smples
Daug
λEm
λEx-C
MCR
ALS
SEx-C
ST Em
Emission
Wavelength (nm)
SEx1-C1=C1 ⊗ SEx1
SEx1-C1 new
SVD
1st component
Concentration
Excitation

Figure 2

3 components

4 components

5 components

6 components

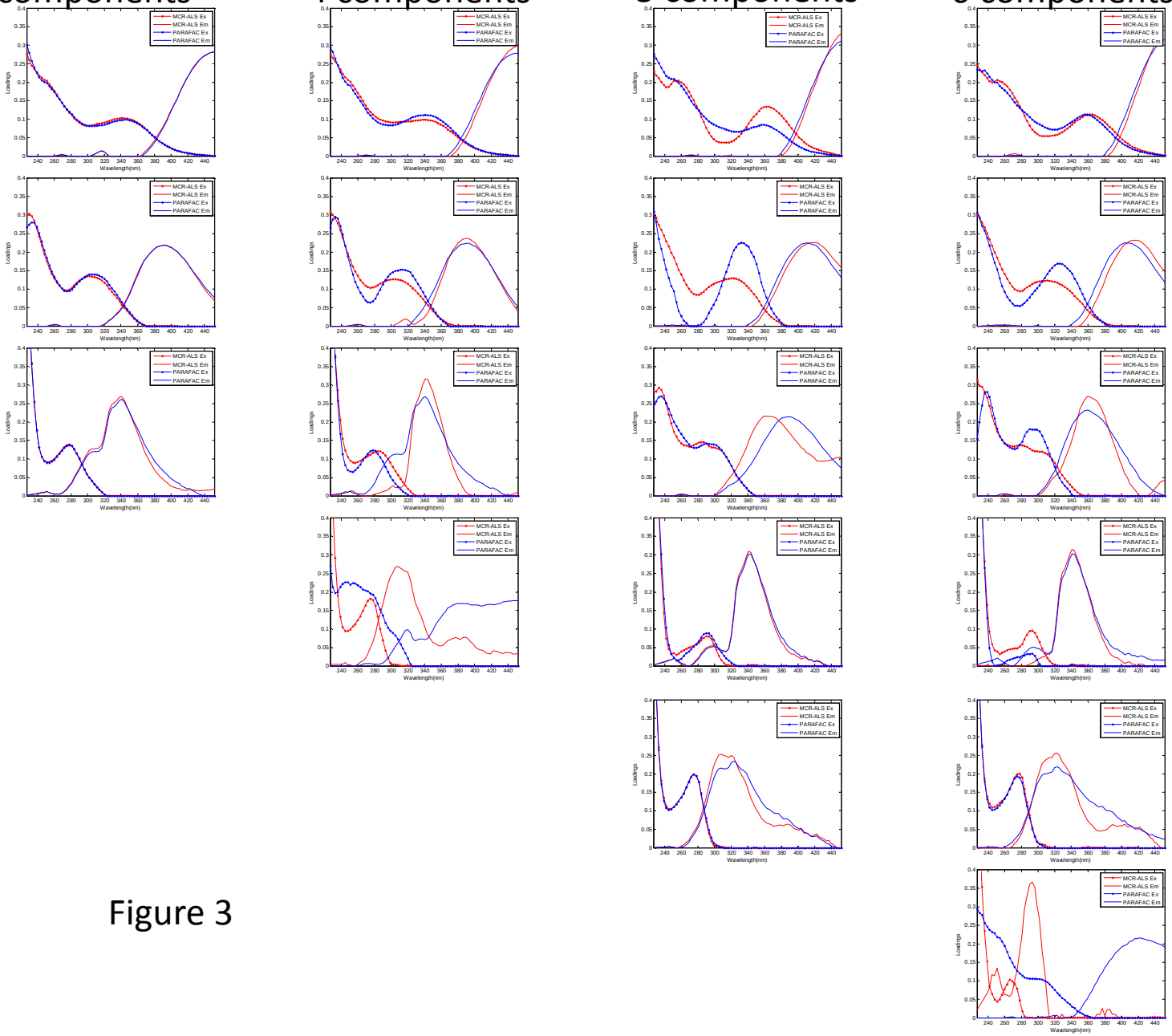


Figure 3

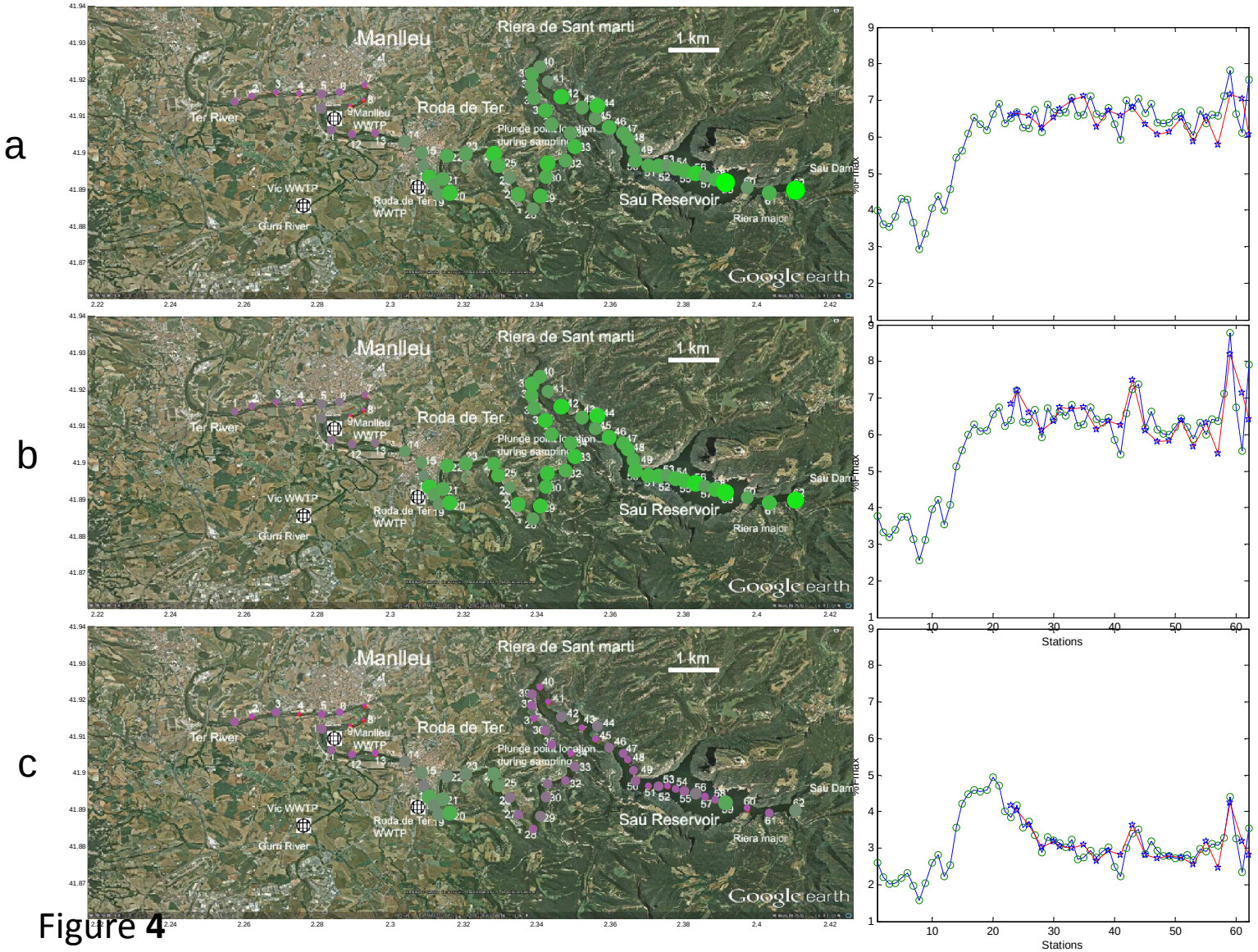
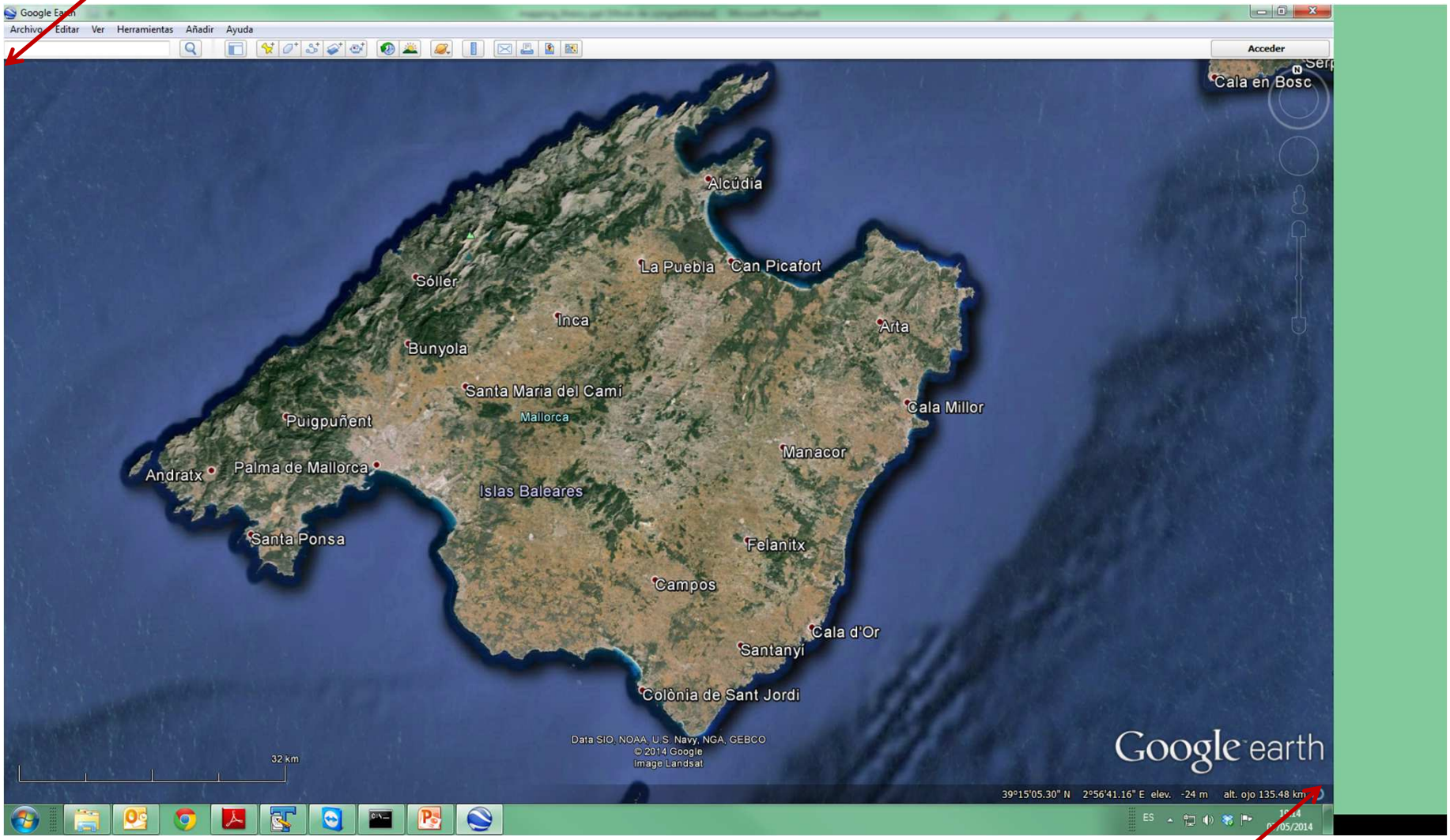

a
b
c
Manlleu
Riera de Sant marti
1 km
Ter River
Manlleu WWTP
Roda de Ter
Plunge point location during sampling
Vic WWTP
Roda de Ter WWTP
Gurri River
Sau Reservoir
Riera major
Sau Dam
Google earth
%FHMax
Stations

Figure **4**

Lat 40.01 N, Lon 2.09 E



Lat 39.09 N, Lon 3.56 E

Producing map images (ex. from google earth) and georefencing them

Make a map of Cap Salines in Mallorca using Google Earth and store in an image jpg file capsalines.jpg

```
>> A=imread('capsalines2.jpg','jpg');
>> size(A)
ans =
         758        1280           3

>> latmin=39.1538
>> latmax=39.1612
>> lonmin=3.0221
>> lonmax=3.0327
>> dlon=(lonmax-lonmin)/1280
>> dlat=(latmax-latmin)/758

refvec=[1/dlon,39.1612,3.0327]
refvec =
  1.0e+005 *
    1.2075    0.0004    0.0000
```

```
>> R = MAKEREFMAT(39.15, 3.02,
dlon, dlat)
R =
         0    0.0000
    0.0000         0
   39.1500    3.0200
```

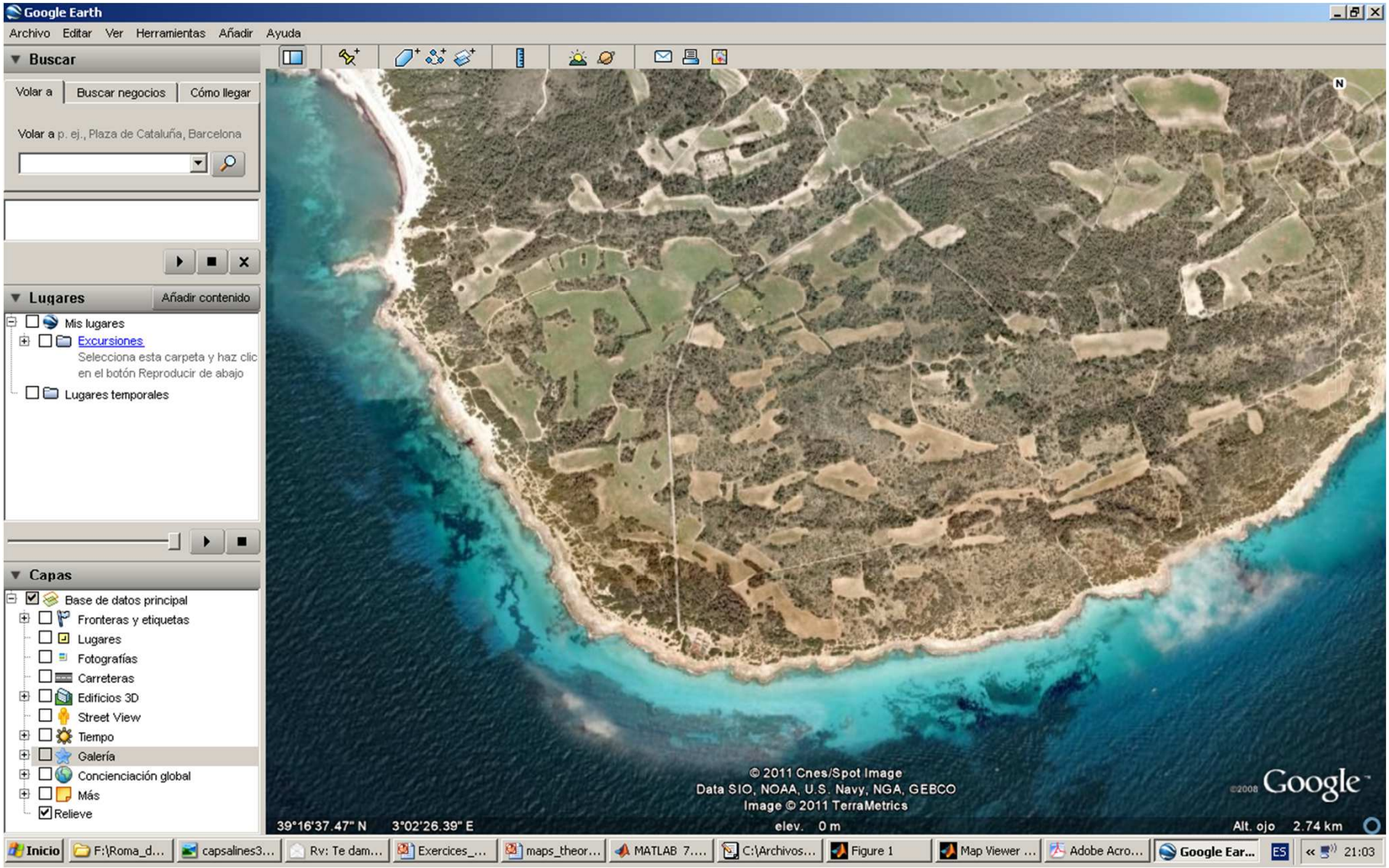
AI=A(end:-1:1,:,:);

mapview

Check for the correct georeferences

Save image as a GeoTiff
It creates two files:
one .tiff and another .tfw

Now it can already be read directly as a GeoTiff referecned image for further work



Google Earth

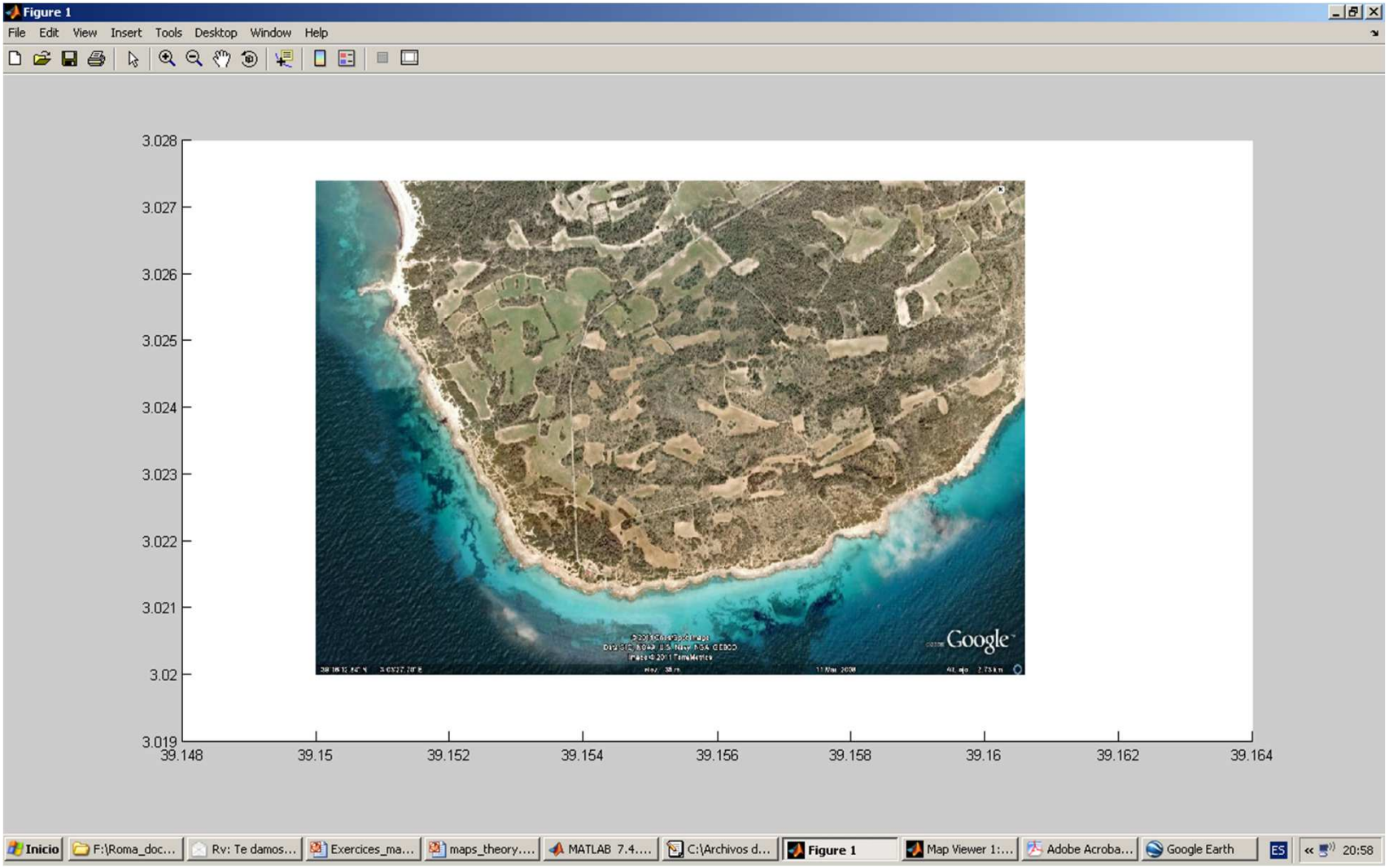
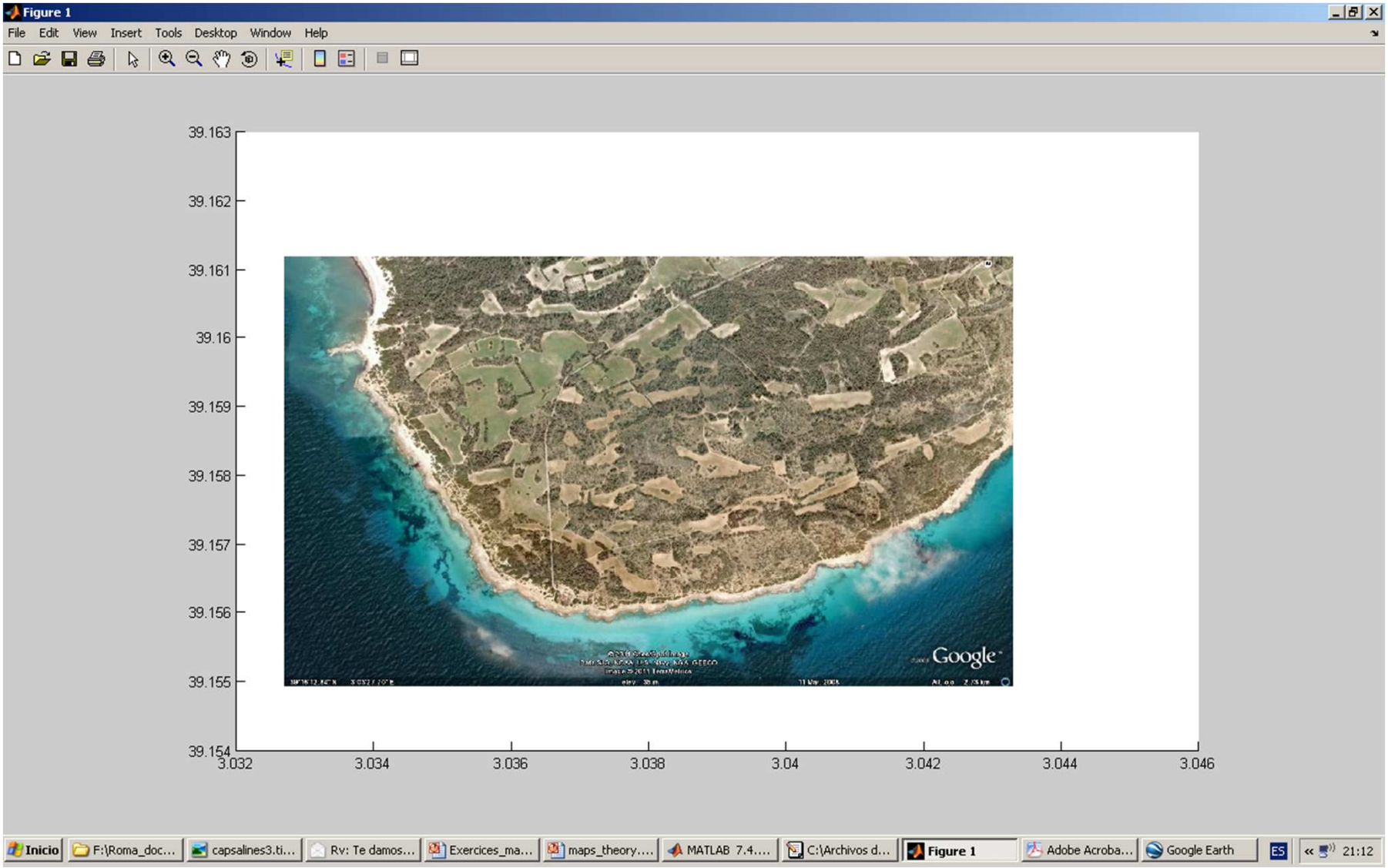
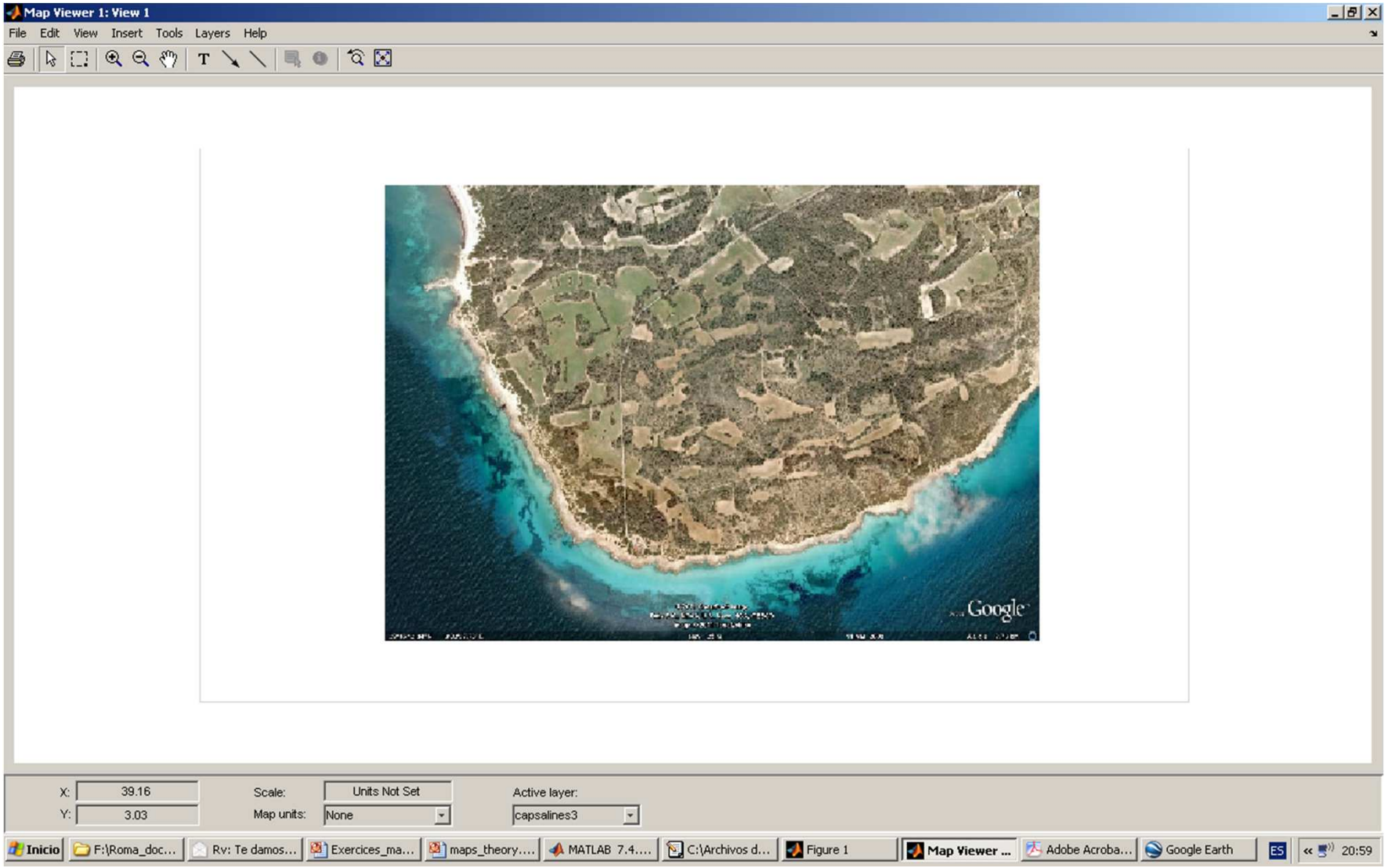


Figure 1
File Edit View Insert Tools Desktop Window Help
3.028
3.027
3.026
3.025
3.024
3.023
3.022
3.021
3.02
3.019
39.148
39.15
39.152
39.154
39.156
39.158
39.16
39.162
39.164
Google
Inicio
F:\Roma_doc...
Rv: Te damos...
Exercices_ma...
maps_theory....
MATLAB 7.4....
C:\Archivos d...
Figure 1
Map Viewer 1:...
Adobe Acroba...
Google Earth
ES
20:58

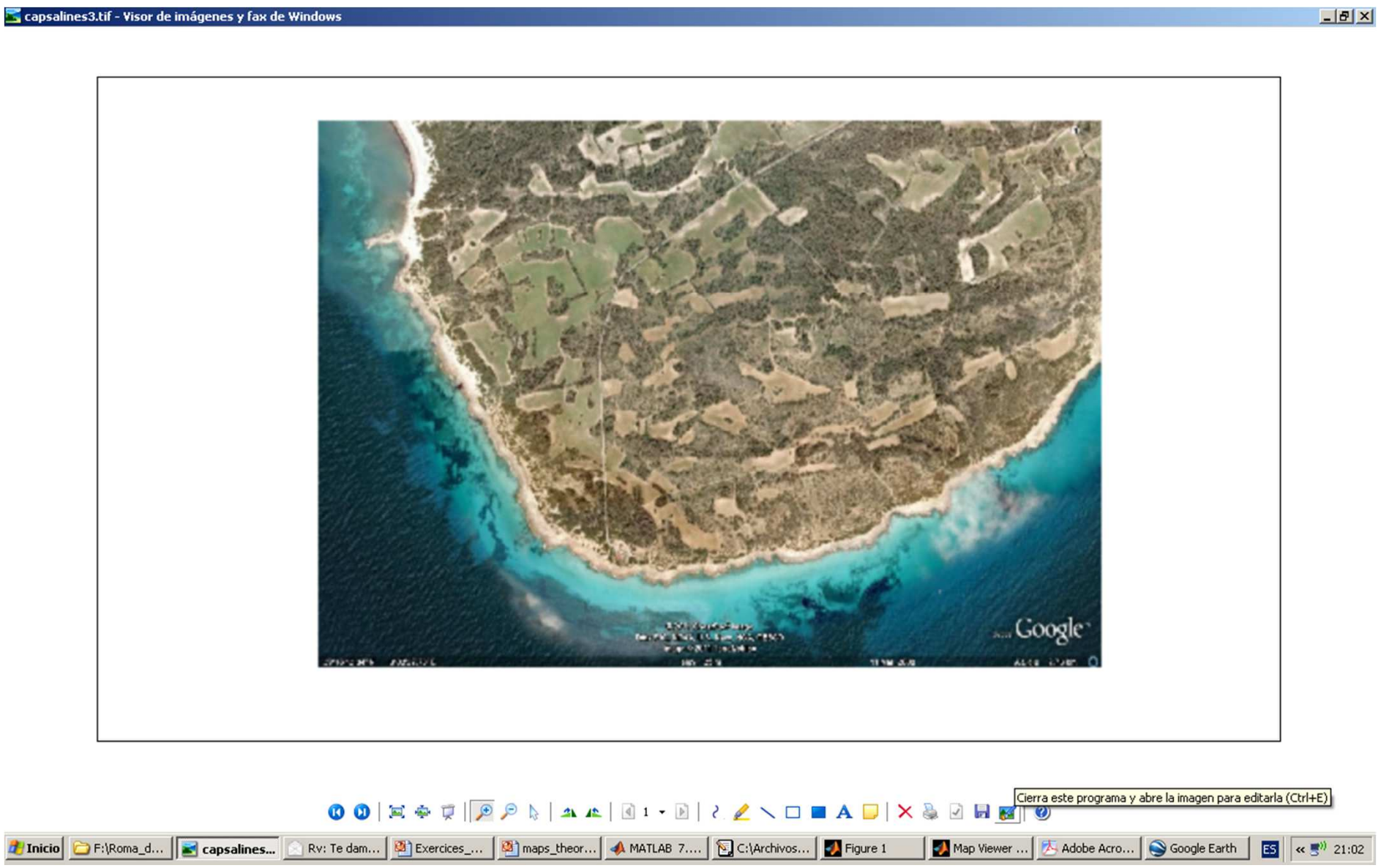
geoshow(AI,R)




```
>> refvec=[1/dlon,39.1612,3.0327]
refvec =   1.0e+005 * 1.2075    0.0004    0.0000
>> geoshow(AI,refvec)
```


Map Viewer 1: View 1
File Edit View Insert Tools Layers Help
Google
X: 39.16
Y: 3.03
Scale: Units Not Set
Map units: None
Active layer: capsalines3
Inicio
F:\Roma_doc...
Rv: Te damos...
Exercices_ma...
maps_theory....
MATLAB 7.4....
C:\Archivos d...
Figure 1
Map Viewer ...
Adobe Acroba...
Google Earth
ES
20:59

Map Viewer



capsalines.tif, capsalines.tfw

Mapping

Mapping

Introduction to MATLAB mapping Toolbox
Data Models
Map projections
MATLAB mapping Toolbox and georeferencing
Introduction to M_map Toolbox

Laboratory exercises:

Plotting images
Google map loader
Getting satellite data from internet
Representing results from a monitoring campaign

Romà Tauler (IDAEA, CSIC, Barcelona)

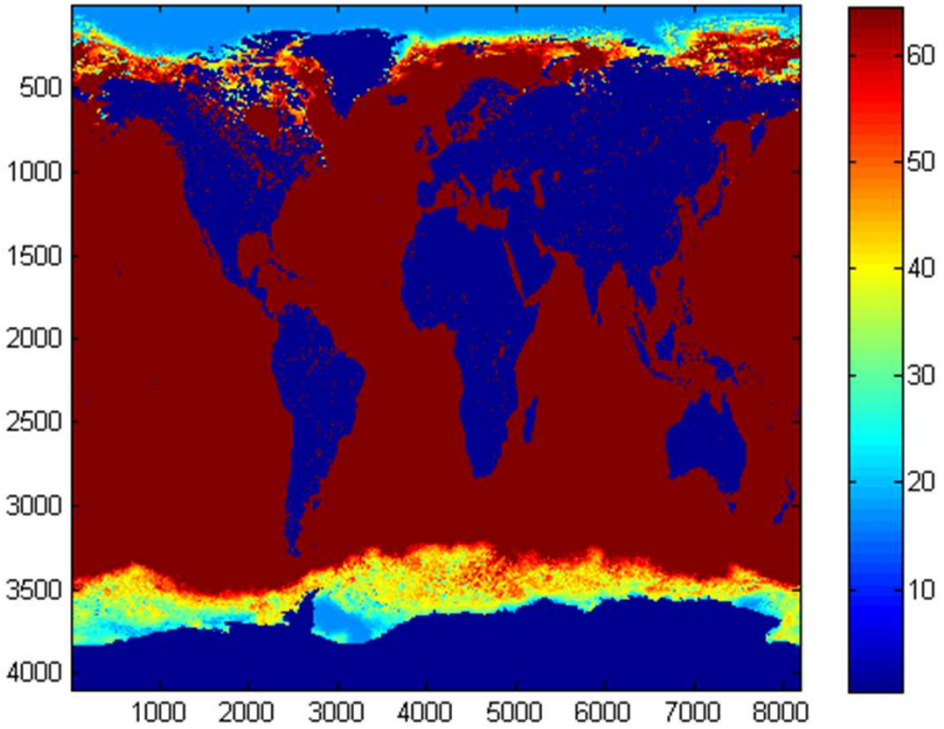
Geeting satellite data from internet

Examples of satellite data manipulation
1. Global SST (or any variable on a global Lat/Long grid)
% NOAA/NASA Pathfinder AVHRR SST product
% http://podaac.jpl.nasa.gov/sst/

For example:
in subdirectory nasa from lab_maps
bsst2007.hdf
Use import bizard of MATLAB
Choose bsst
It is a matrix of dimensions 4096 819
It can be represented directly using ir

image(bsst)

But the units are not correct!



Georeferencing the map

To calculate the geographic location of a specific pixel use the following equations for 9 km equal-angle imagery (other resolutions will be similar):

longitude = (-180. + (x * dx)) + (dx/2)
latitude = (90. - (y * dy)) - (dy/2)

where for 9km resolution: x = grid point in x-direction (0-4095); dx = grid x-direction spacing (360 deg / 4096); y = grid point in y-direction (0-2047); and dy = grid y-direction spacing (180 deg / 2048.)

for 5.0 4 km resolution: x = grid point in x-direction (0-8191); dx = grid x-direction spacing (360 deg / 8192.); y = grid point in y-direction (0-4095); and dy = grid y-direction spacing (180 deg / 4096.)

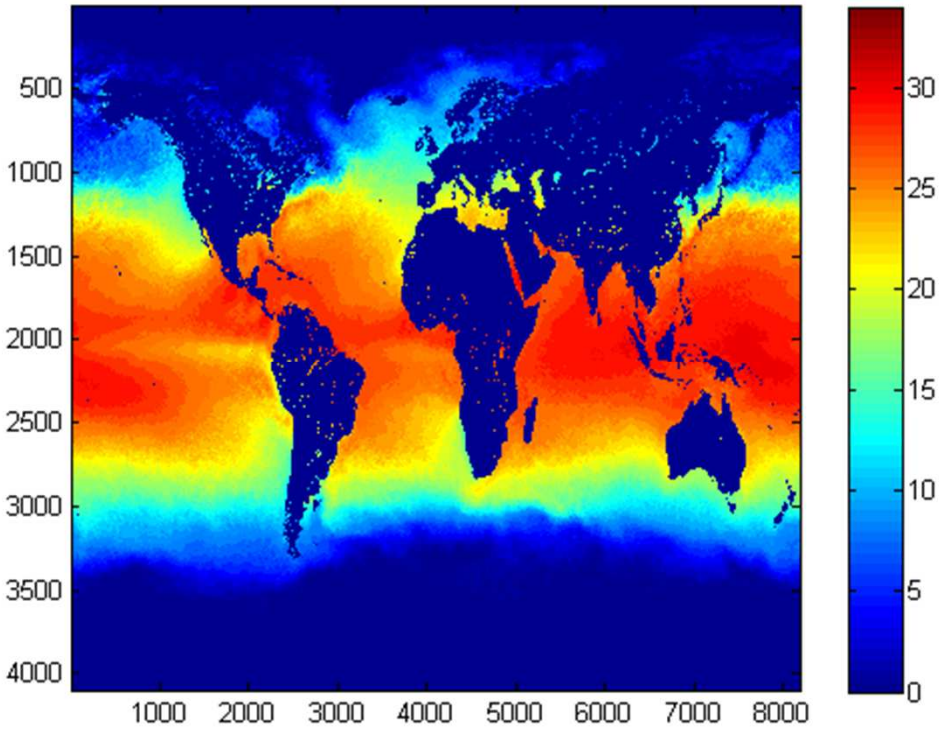
```
>> x=0:8192;
>> y=0:4095;
>> dx=360/8192;
>> dy=180/4096;
>> latitude = (90. - (y * dy)) - (dy/2);
>> longitude = (-180. + (x * dx)) + (dx/2);
```

Data Characteristics:
Parameter/Variable:

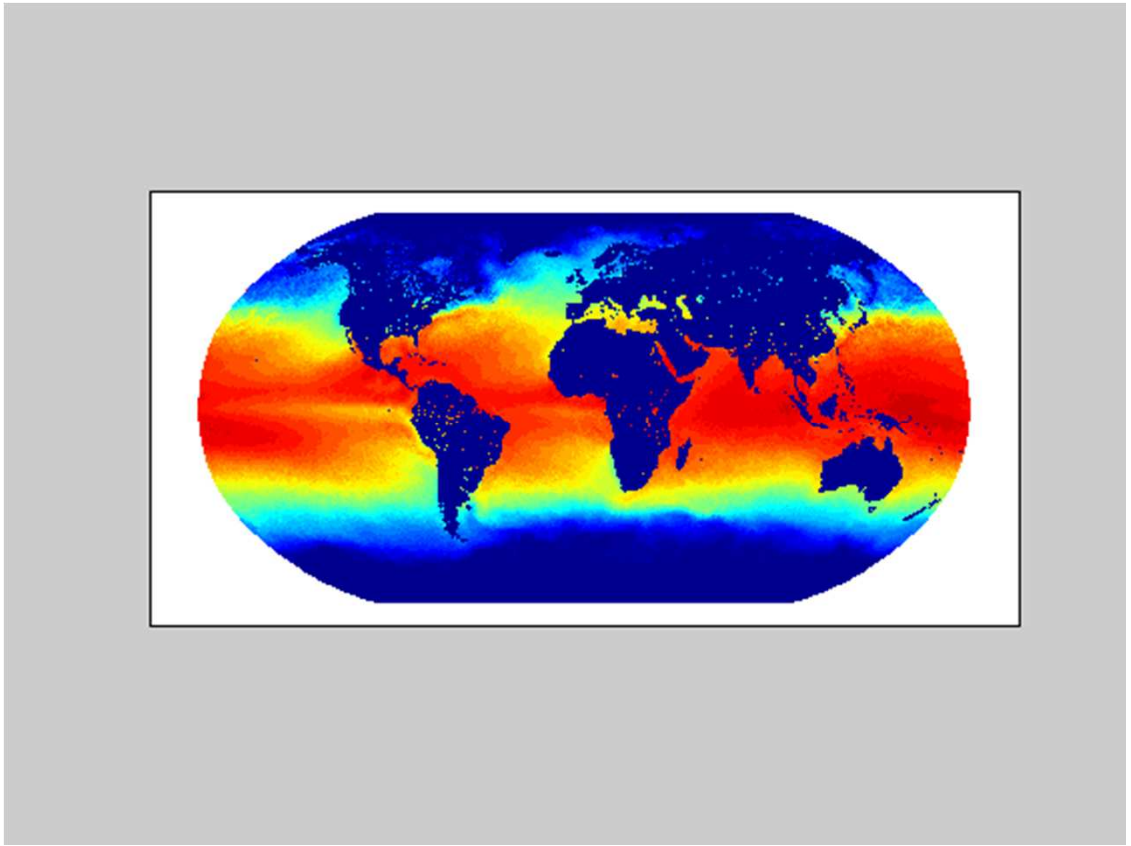
Sea Surface Temperature **Variable Description/Definition:**
SST - temperature of the sea surface.

Units of Measurement:
for version 4.1 and below pixel value with slope of 0.15 and y-intercept of -3.0 to convert to °C.
for version 5.0 pixel value with slope of 0.075 and y-intercept of -3.0 to convert to °C.

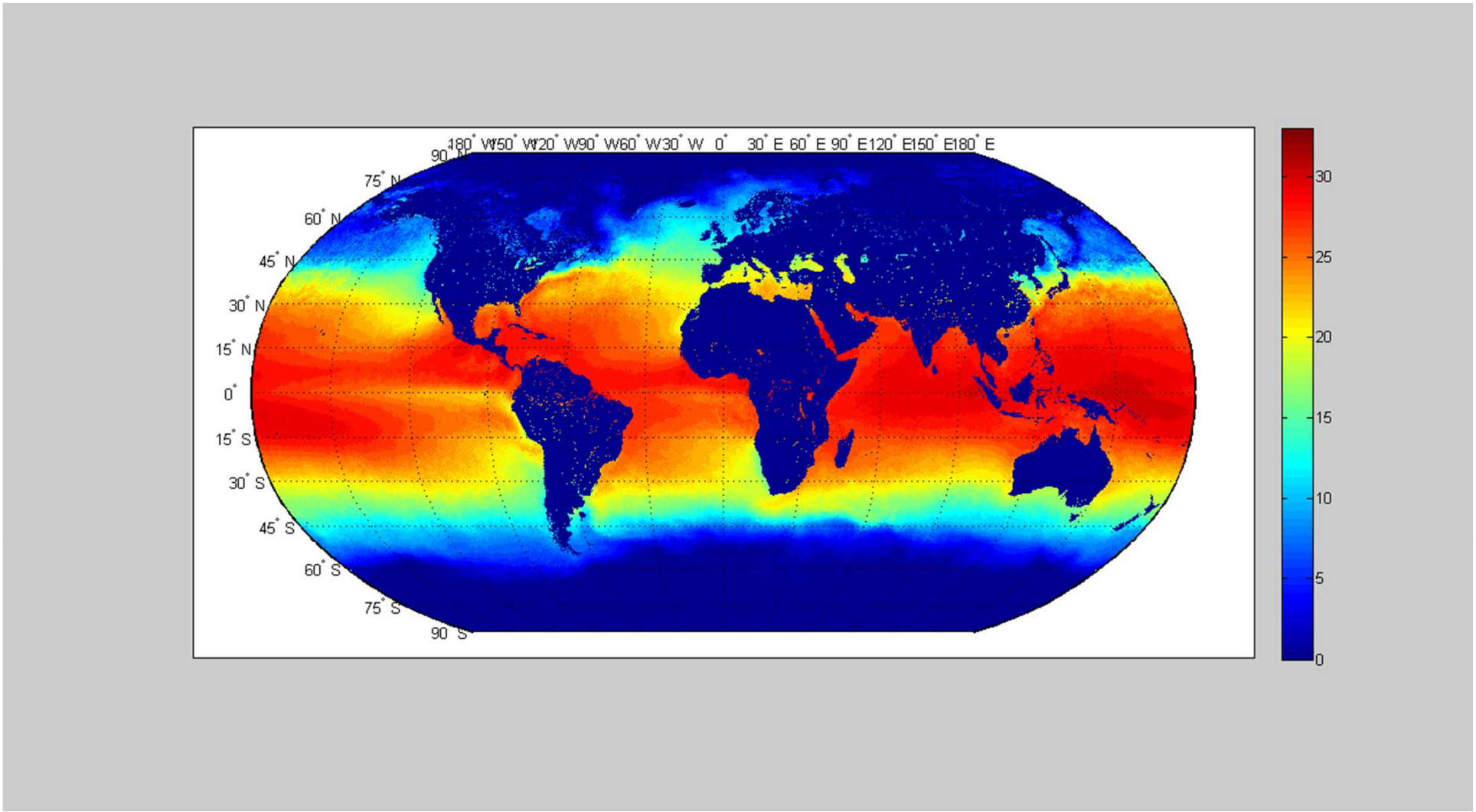
```
>> temp=bsst.*0.075-3;
>> imagesc(temp)
```



```
temp2=temp(1:4:4096,1:4:8192);
dx=360/8192
dy=180/4096
axesm Robinson
temp2=temp(1:4:4096,1:4:8192);
refmat2=makerefmat(-180,90,dx*4,-dy*4)
geoshow(temp2,refmat2,'DisplayType','texturemap')
```



gridm


180° W150° W120° W90° W60° W30° W 0° 30° E 60° E 90° E120° E150° E180° E
90° N
75° N
60° N
45° N
30° N
15° N
0°
15° S
30° S
45° S
60° S
75° S
90° S
30
25
20
15
10
5
0

Mapping

Mapping

Introduction to MATLAB mapping Toolbox
Data Models
Map projections
MATLAB mapping Toolbox and georeferencing
Introduction to M_map Toolbox

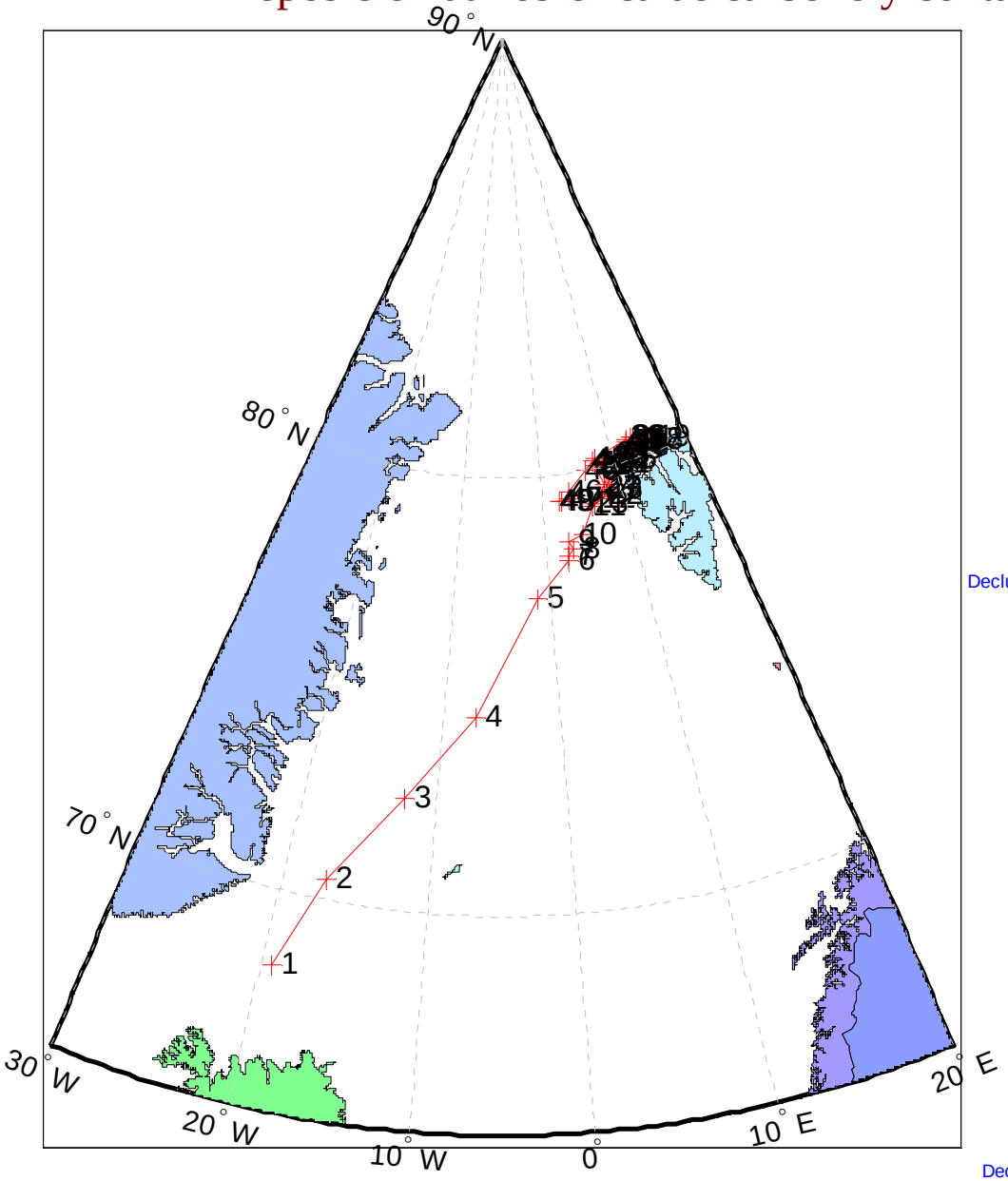
Laboratory exercises:

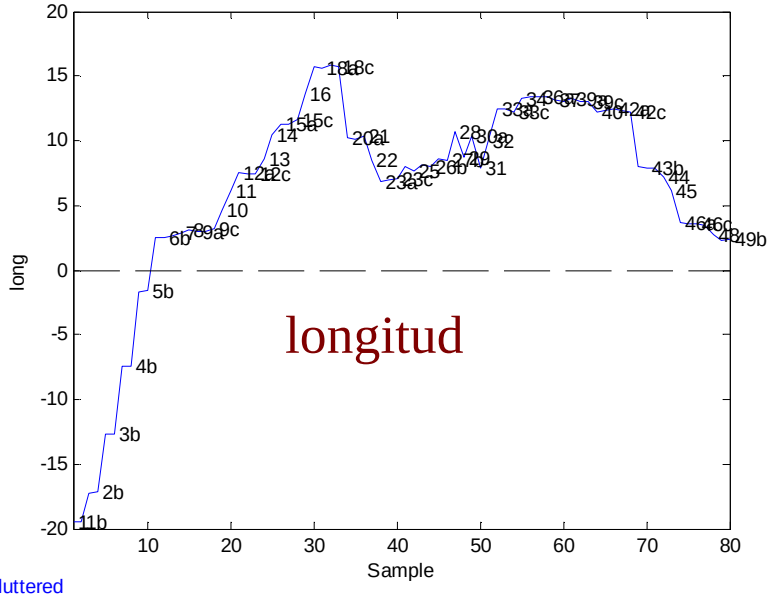
Plotting images
Google map loader
Getting satellite data from internet
Representing results from a monitoring campaign

Romà Tauler (IDAEA, CSIC, Barcelona)

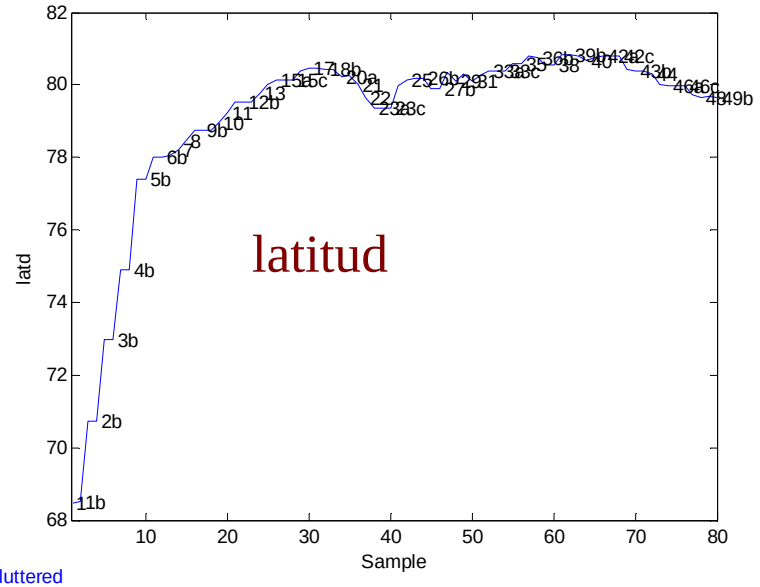
PROYECTO ATOS (Julio 2007)

Deposición atmosférica de carbono y contaminates a los ecosistemas polares





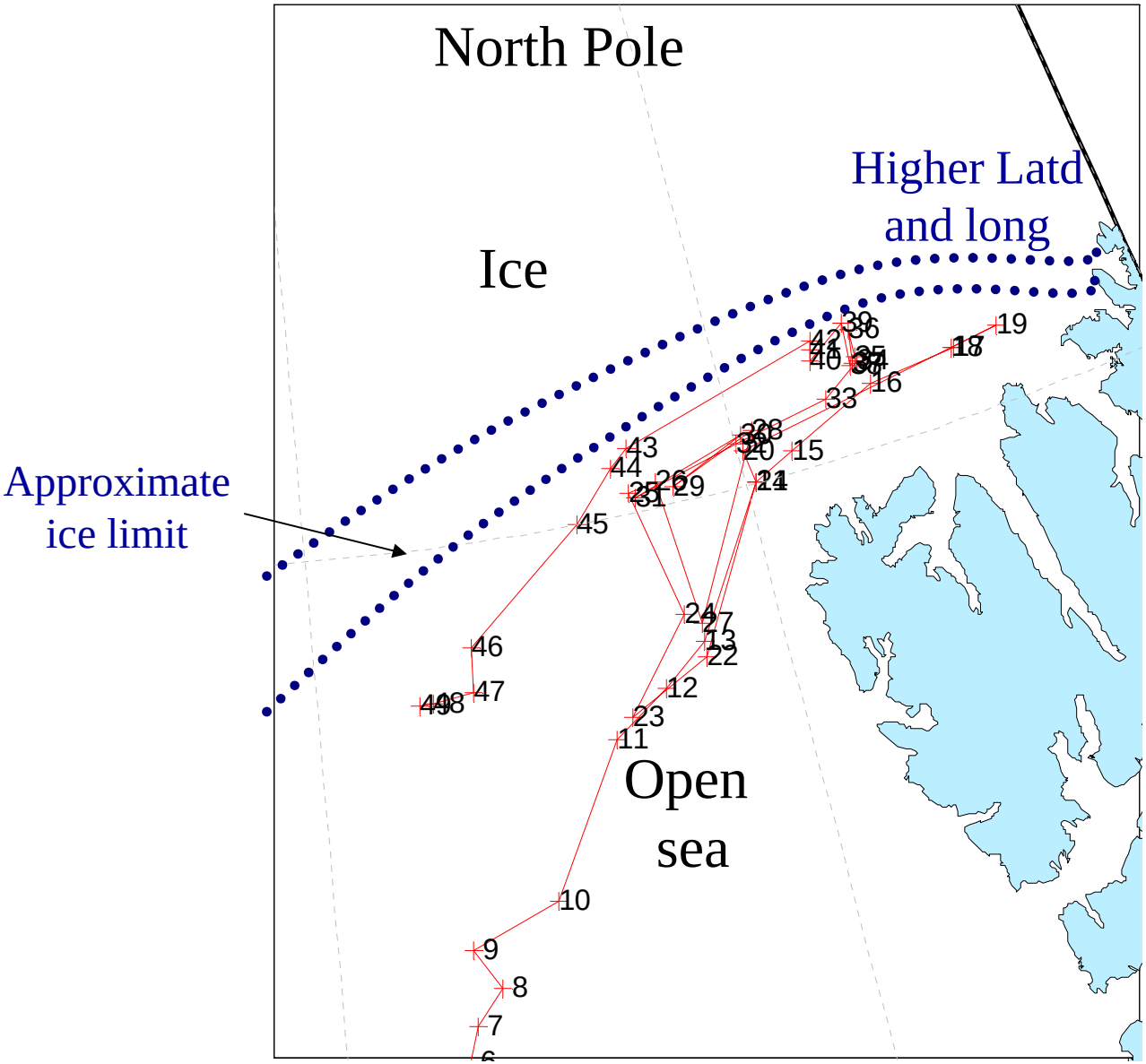
Decluttered



Decluttered

PROYECTO ATOS (Julio 2007)

Deposición atmosférica de carbono y contaminates a los ecosistemas polares



group tasks:

POP analysis in samples:
-air and aerosols
-sea-water
-Ice
-Fito- and zooplancton

Experiments:
-fotodegradation
-biodegradation (zoo)
-accumulation
-toxicity

Method Implementation
-SBSE+TD+GC-MS

Data evaluation and Integration

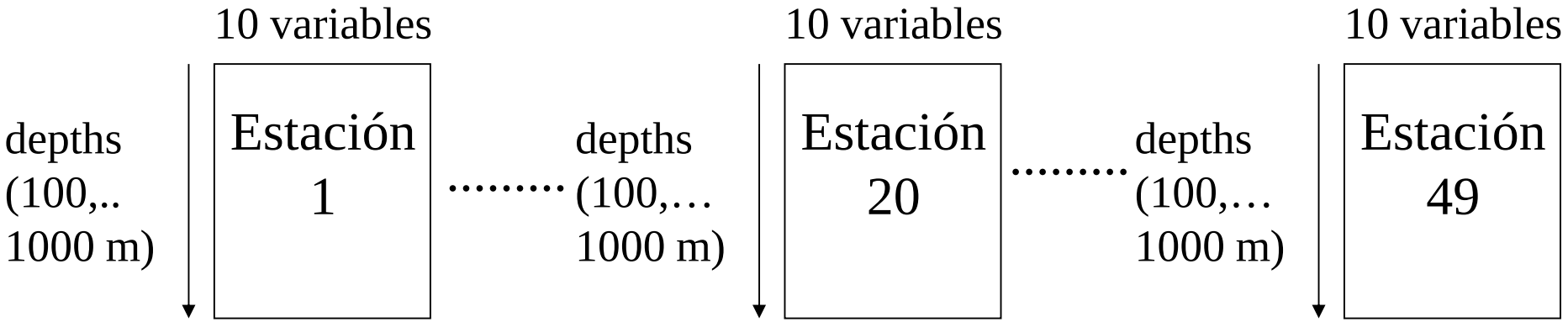
Example: CTD data evaluation and Integration

Buque Hespérides

CTD= Conductivity, Temperature, Depth and other sensors: salt and dissoved O_2 conc., beam light transmission, **fluorescence**, turbidimetry

Example: CTD data evaluation and Integration

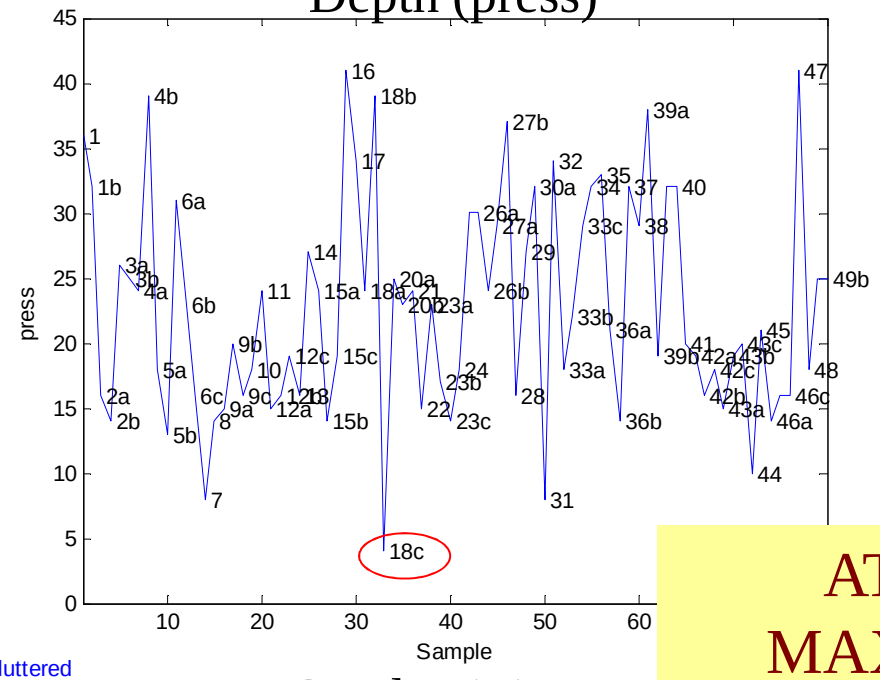
El CTD proporciona medidas de 10 variables a diferentes profundidades en cada una de las estaciones de parada del buque



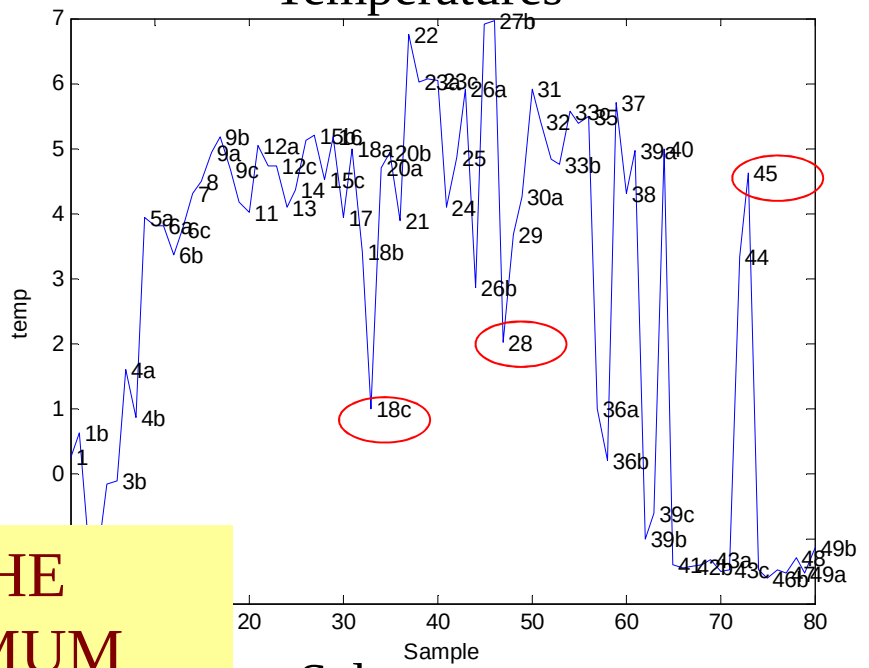
1 'press‘ or depths
2 'temp'
3 'cond'
4 'salt'
5 'oxyg'
6 'btra'
7 'fluo'
8 'turb'
9 'long'
10 'latd'

La adquisición de datos se realiza muy rápidamente de forma continua por lo que se requiere realizar un promediado y filtrado previo de los datos.
Tabla o Matriz final de datos a procesar : **X(53367x10)** provenientes de 81 experimentos CTD (49 estaciones con algunas réplicas y profundidades diferentes, total 80 estac.)

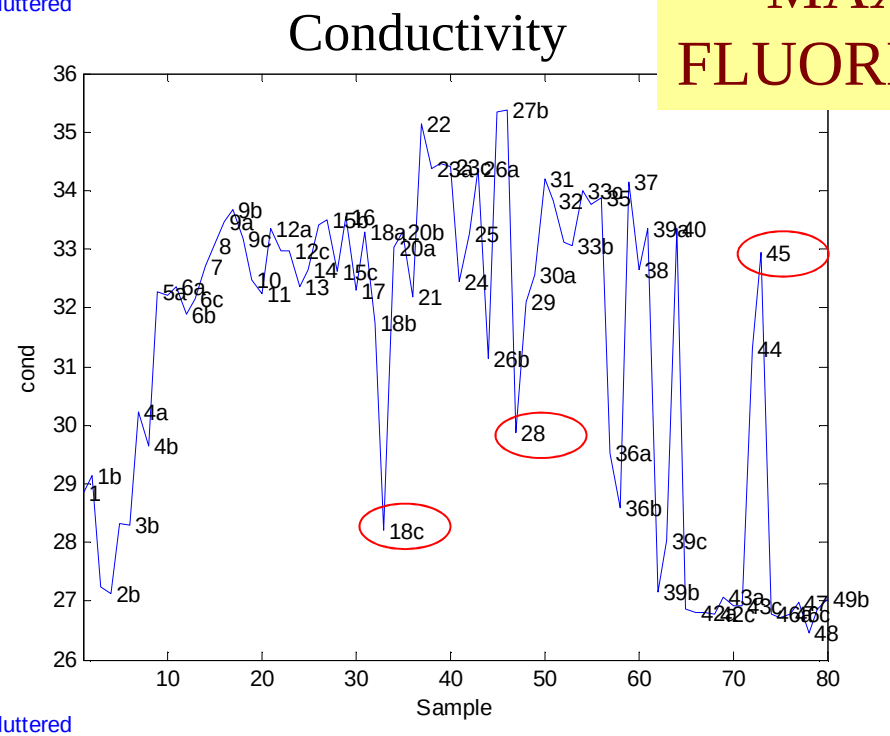
Se seleccionan algunos casos de interés:
1) Matrices $X_1(80,10)$ **$X_{DCM}(80,10)$,** $X_{100}(80,10)$ respectivamente en superfície, al máximo fluor y a 100m prof.)
2) Matrices de fluorescencia **$X_{fluor}(80,200)$**



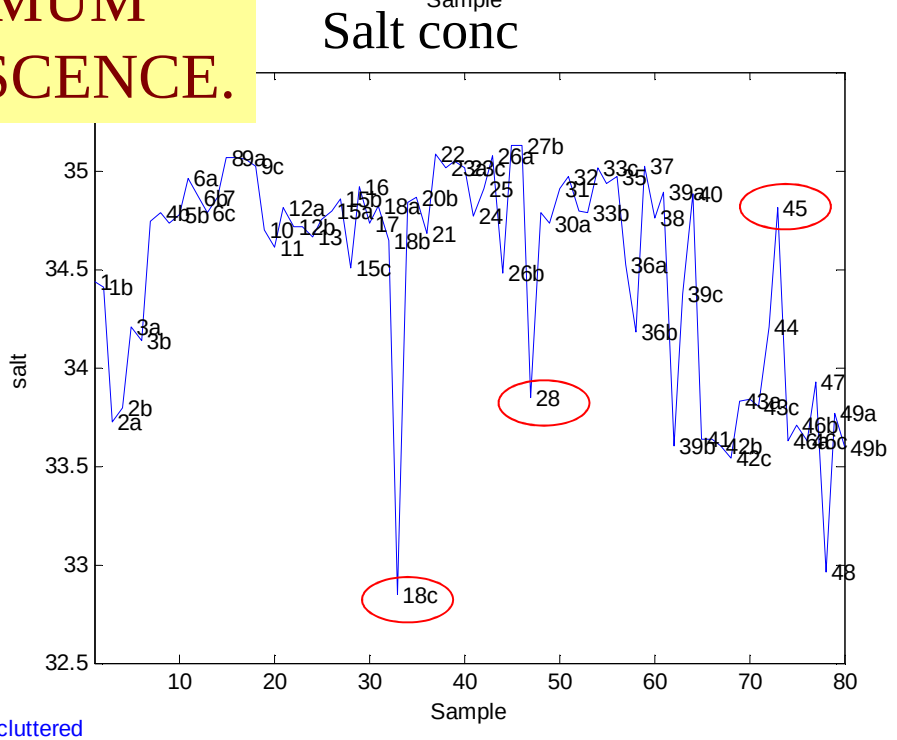
Decluttered



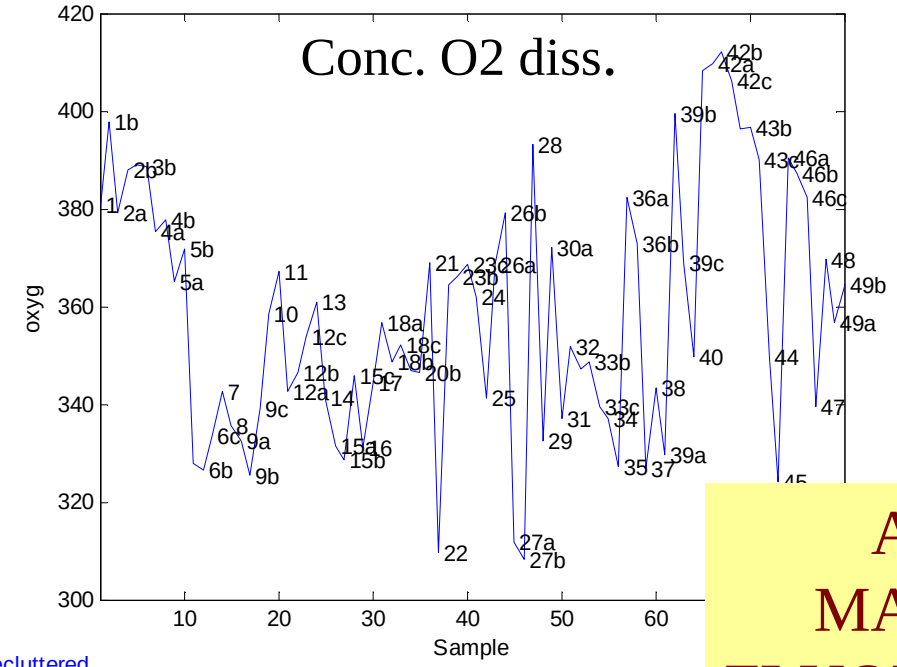
AT THE MAXIMUM FLUORESCENCE.



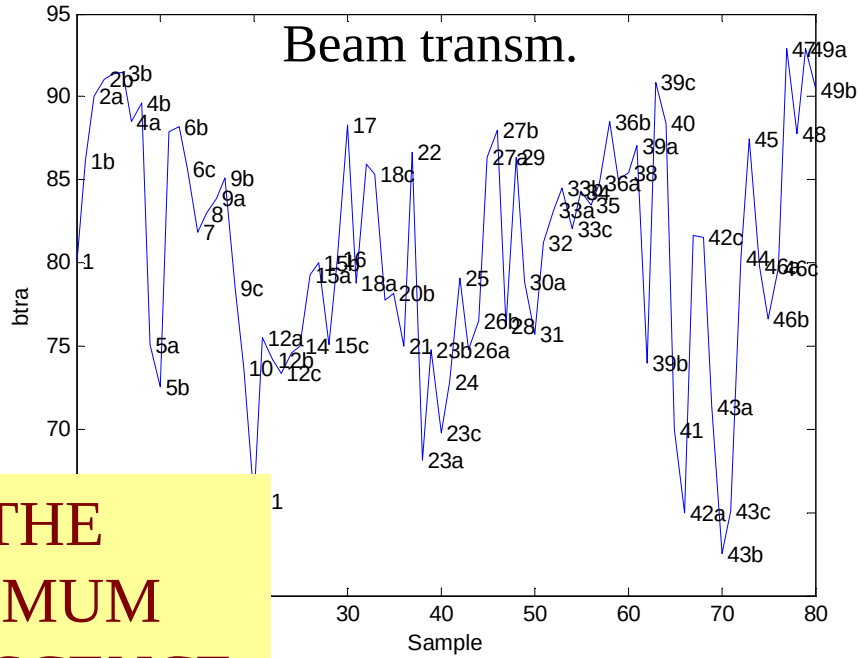
Decluttered



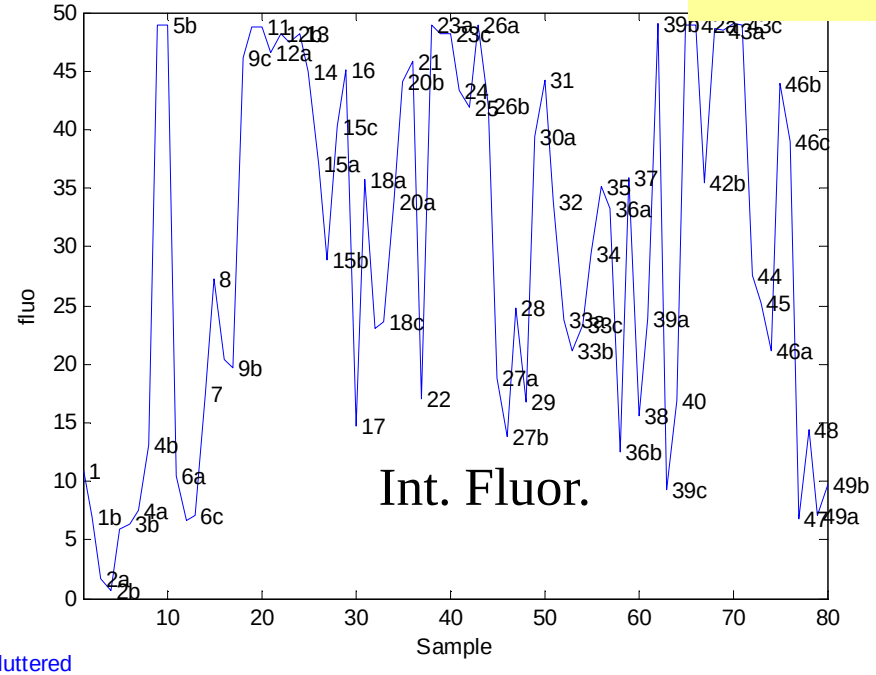
Decluttered



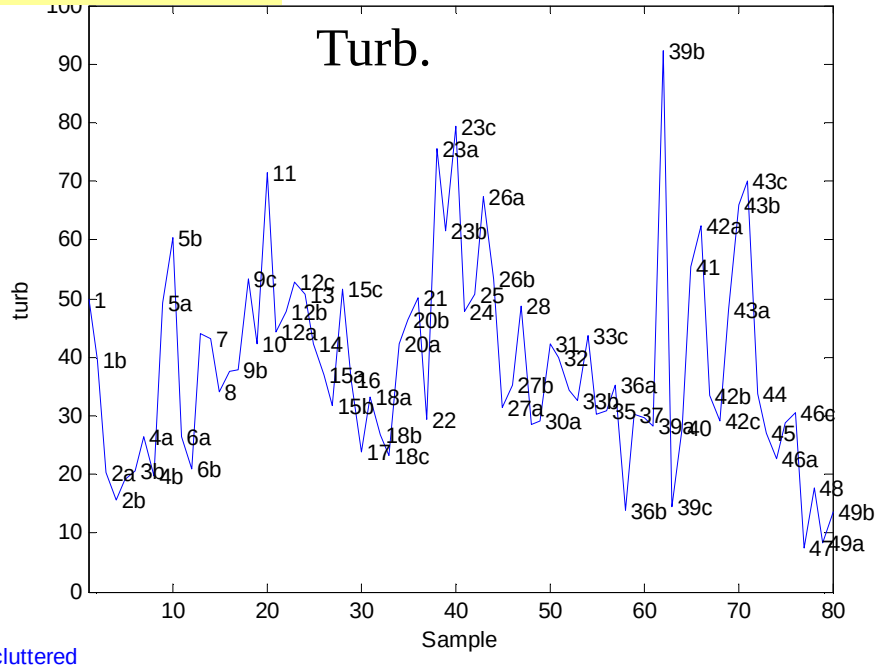
Decluttered



AT THE MAXIMUM FLUORESCENCE.



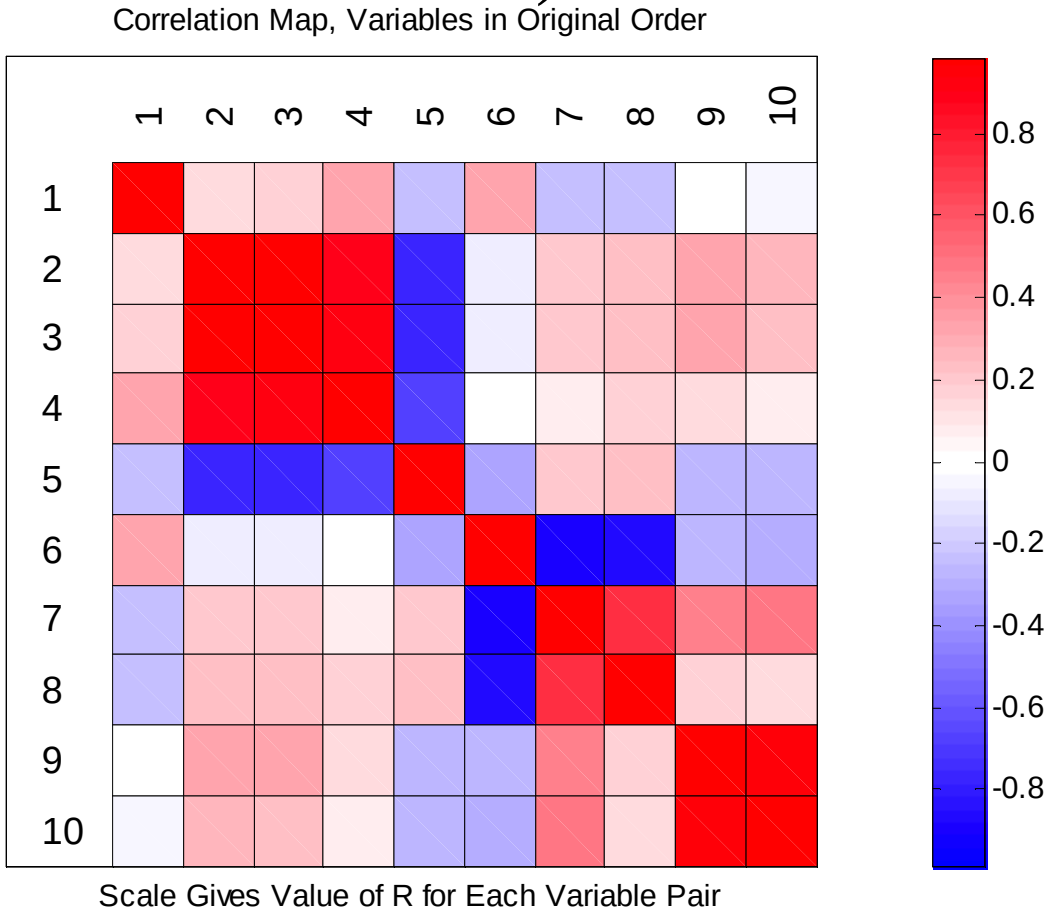
Decluttered



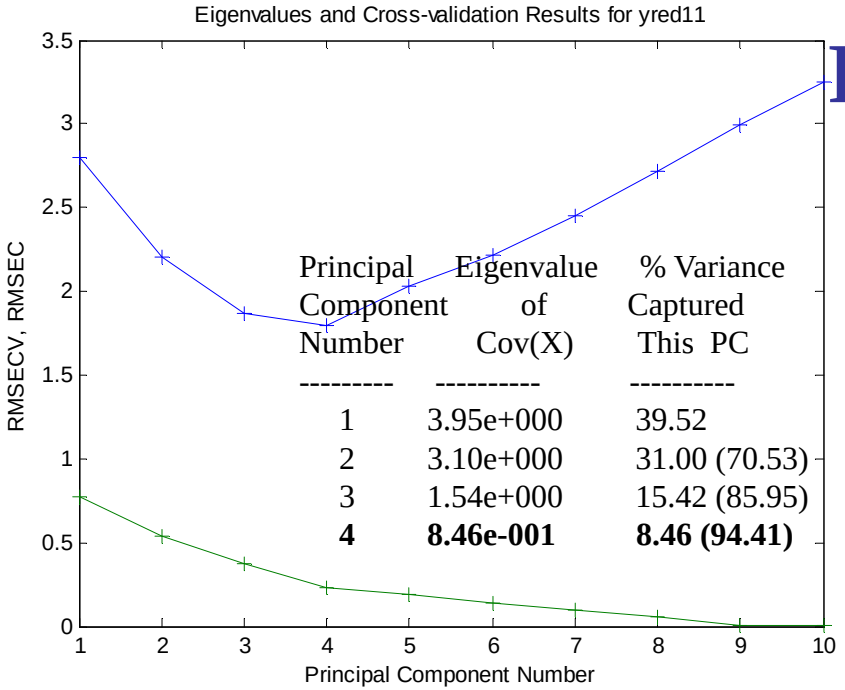
Decluttered

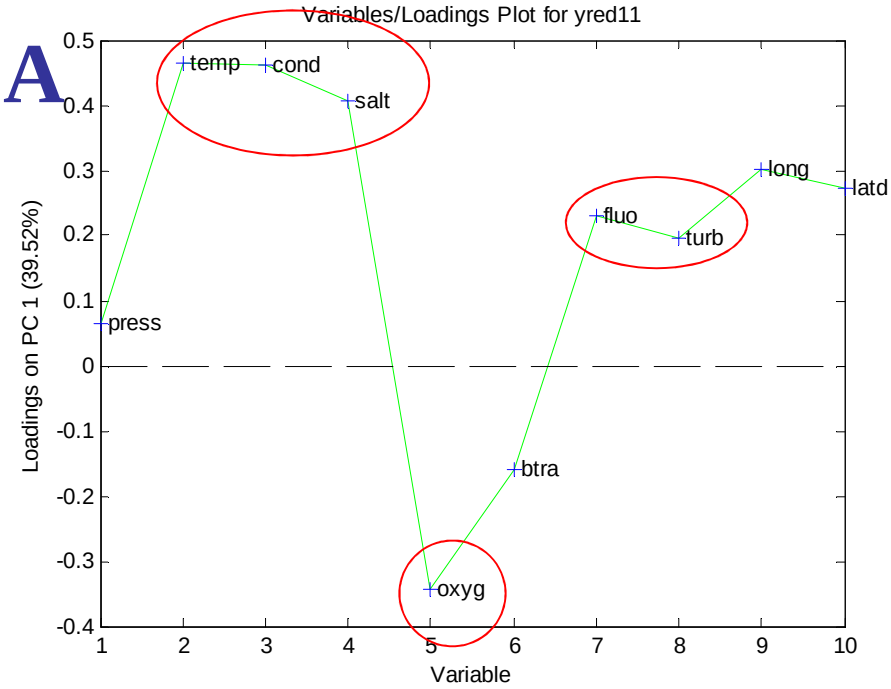
correlation in dcm (máximo de clorofila máximo de fluorescencia)

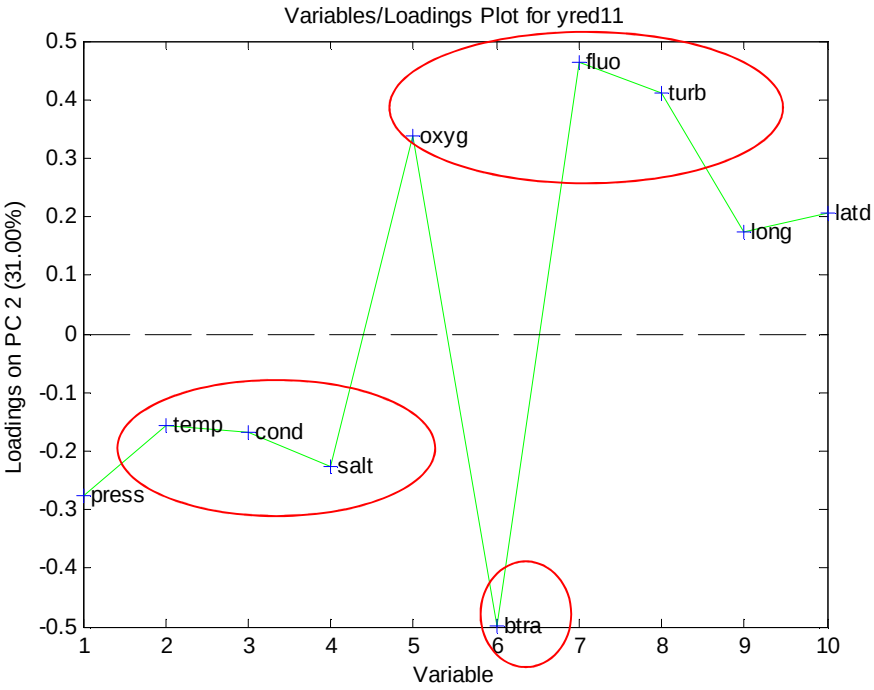
1	'press'
2	'temp'
3	'cond'
4	'salt'
5	'oxyg'
6	'btra'
7	'fluo'
8	'turb'
9	'long'
10	'latd'

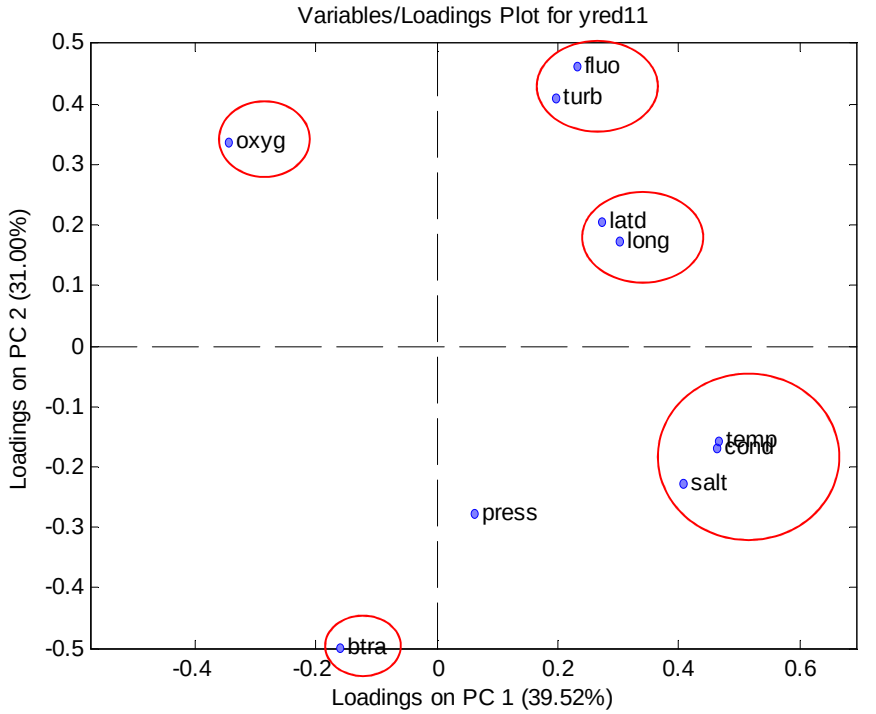


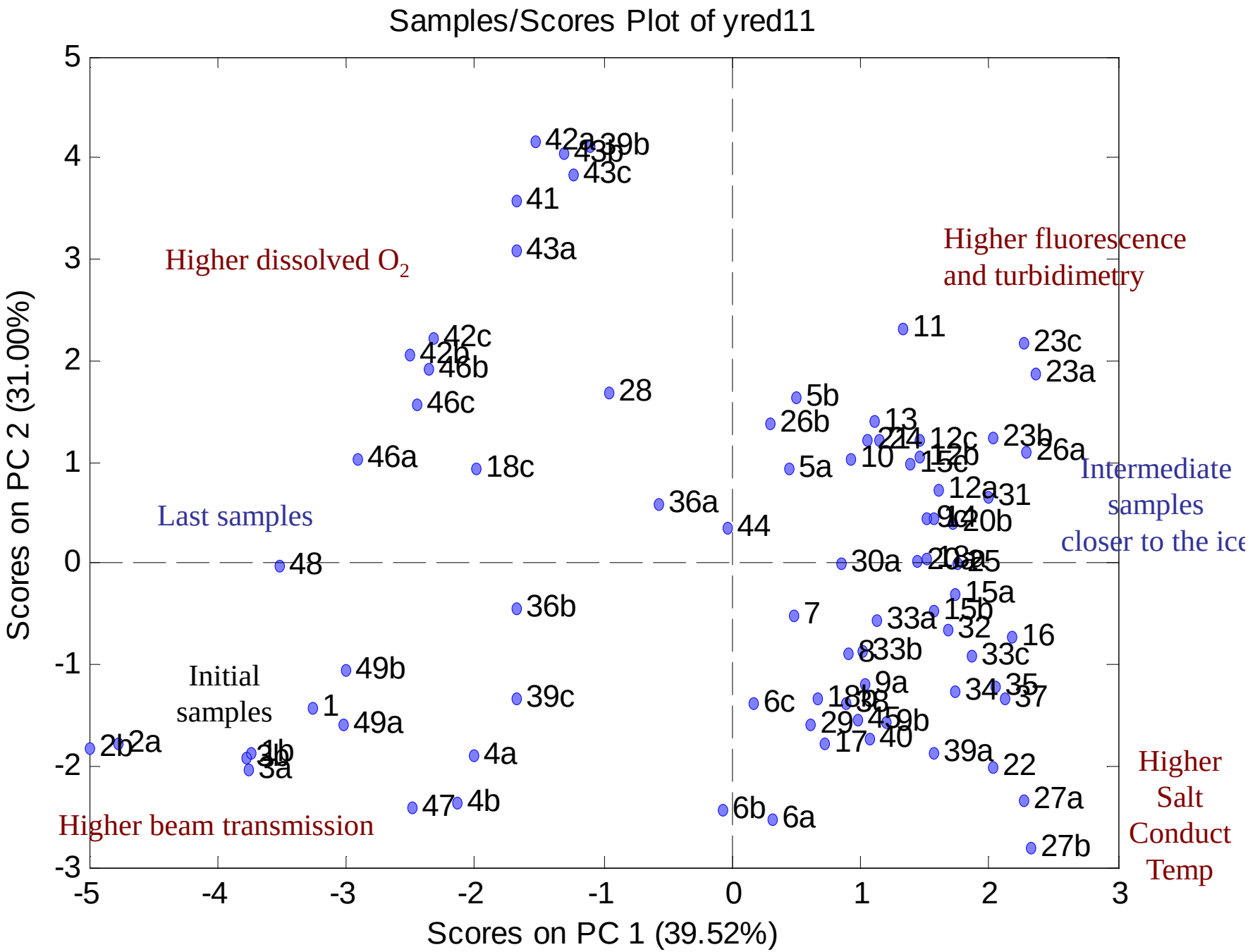
PCA


Eigenvalues and Cross-validation Results for yred11
RMSECV, RMSEC
Principal Component Number
Principal Component Number | Eigenvalue of Cov(X) | % Variance Captured This PC
1 | 3.95e+000 | 39.52
2 | 3.10e+000 | 31.00 (70.53)
3 | 1.54e+000 | 15.42 (85.95)
4 | 8.46e-001 | 8.46 (94.41)


Variables/Loadings Plot for yred11
Loadings on PC 1 (39.52%)
Variable
temp
cond
salt
press
fluo
turb
long
latd
btra
oxyg


Variables/Loadings Plot for yred11
Loadings on PC 2 (31.00%)
Variable
fluo
turb
oxyg
long
latd
temp
cond
salt
press
btra


Variables/Loadings Plot for yred11
Loadings on PC 2 (31.00%)
Loadings on PC 1 (39.52%)
fluo
turb
oxyg
latd
long
temp
cond
salt
press
btra

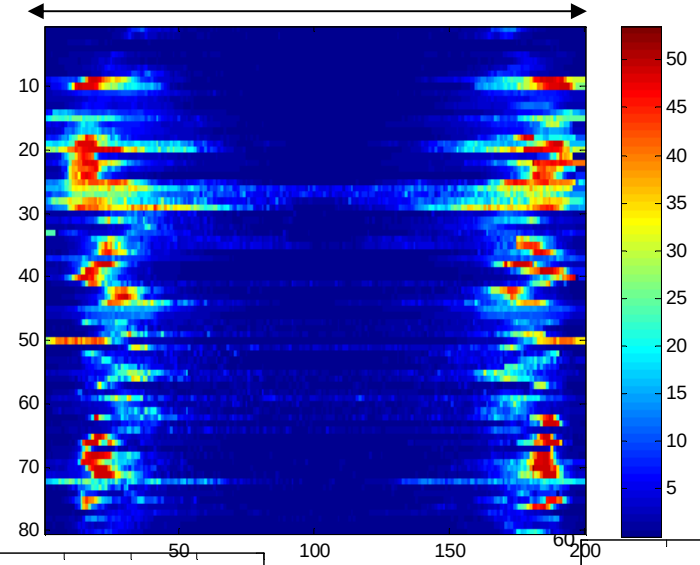

Samples/Scores Plot of yred11
Scores on PC 2 (31.00%)
Scores on PC 1 (39.52%)
Higher dissolved O_2
Higher fluorescence and turbidimetry
Last samples
Intermediate samples closer to the ice
Initial samples
Higher beam transmission
Higher Salt Conduct Temp

Study of the fluorescence at different depths for multiple samples

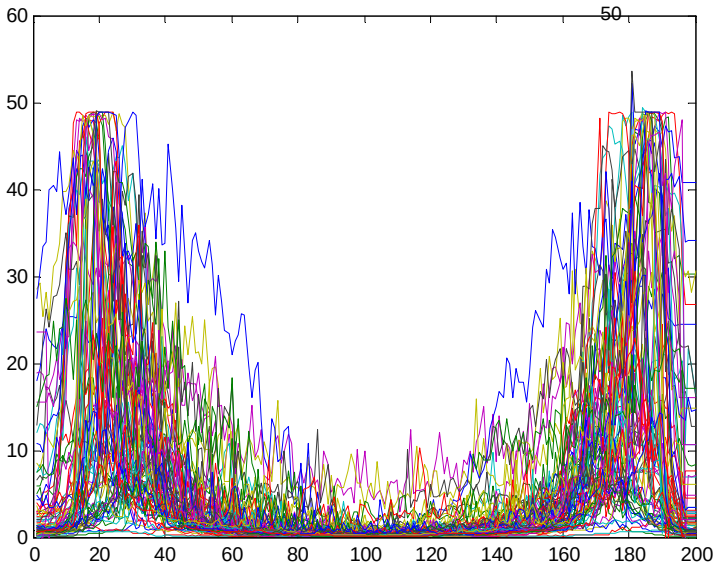
100x2 deepths (CTD down and up)

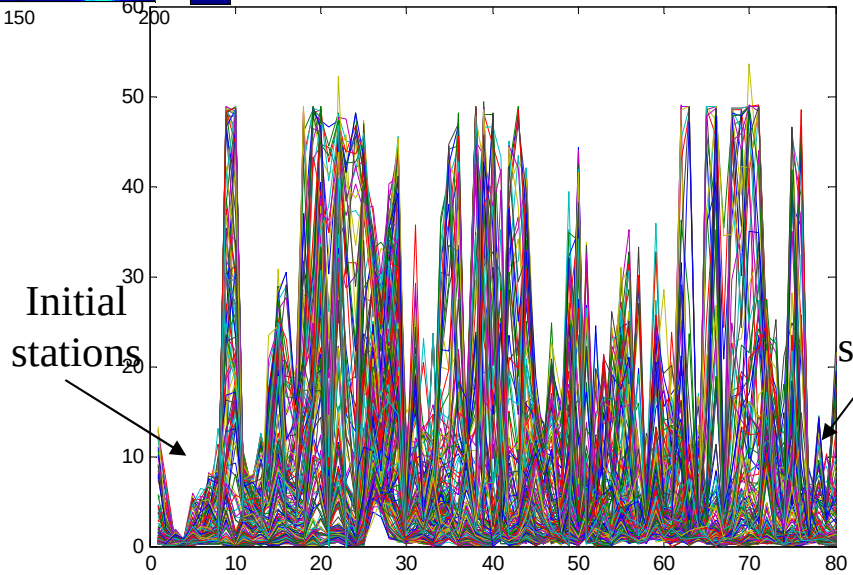
$\mathbf{X}_{fluor}(80,200)$

80 stations

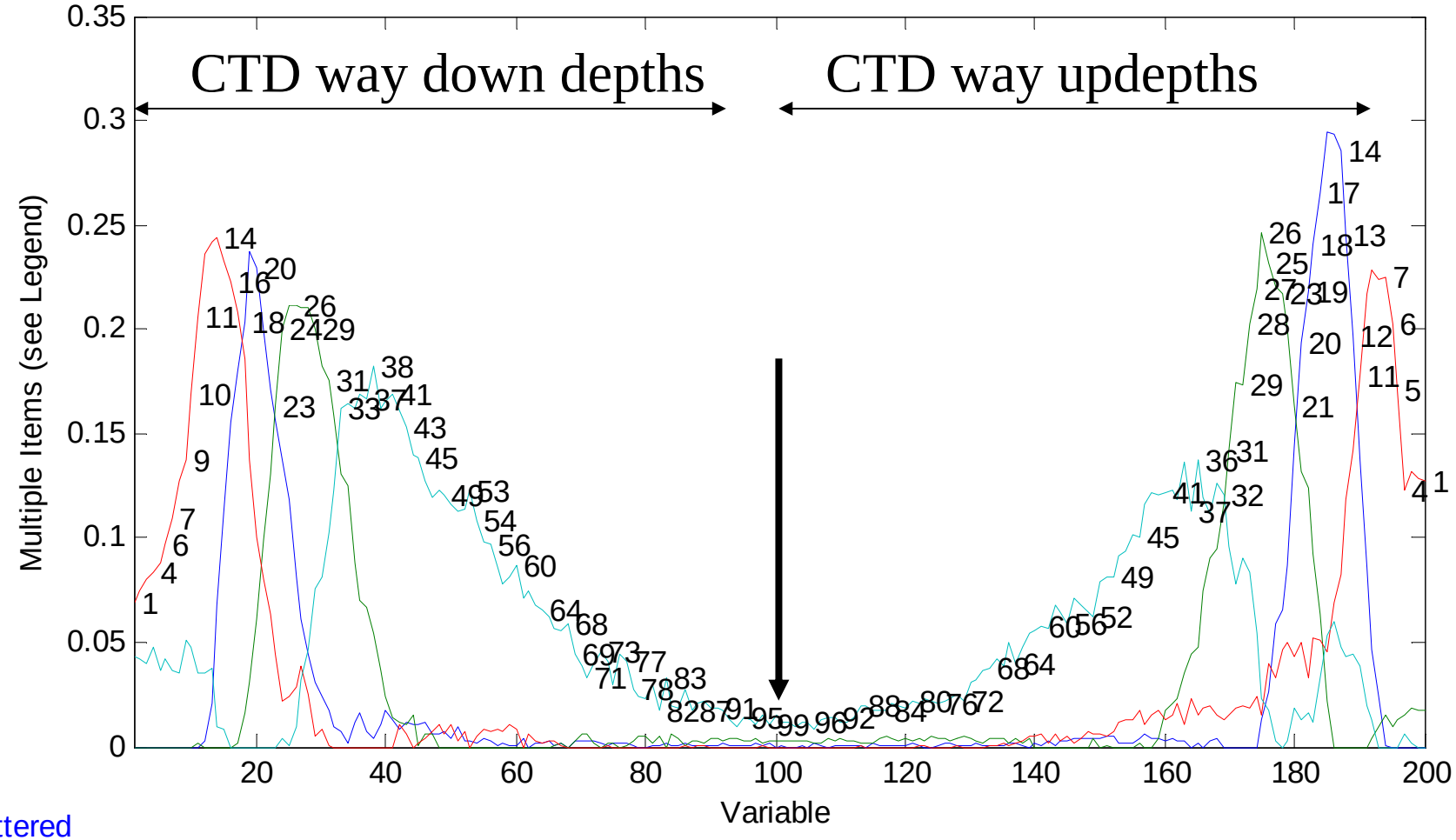


Fluorescence
at 425 nm excitation
and 685 nm emission
(chlorophila fluorescence.)



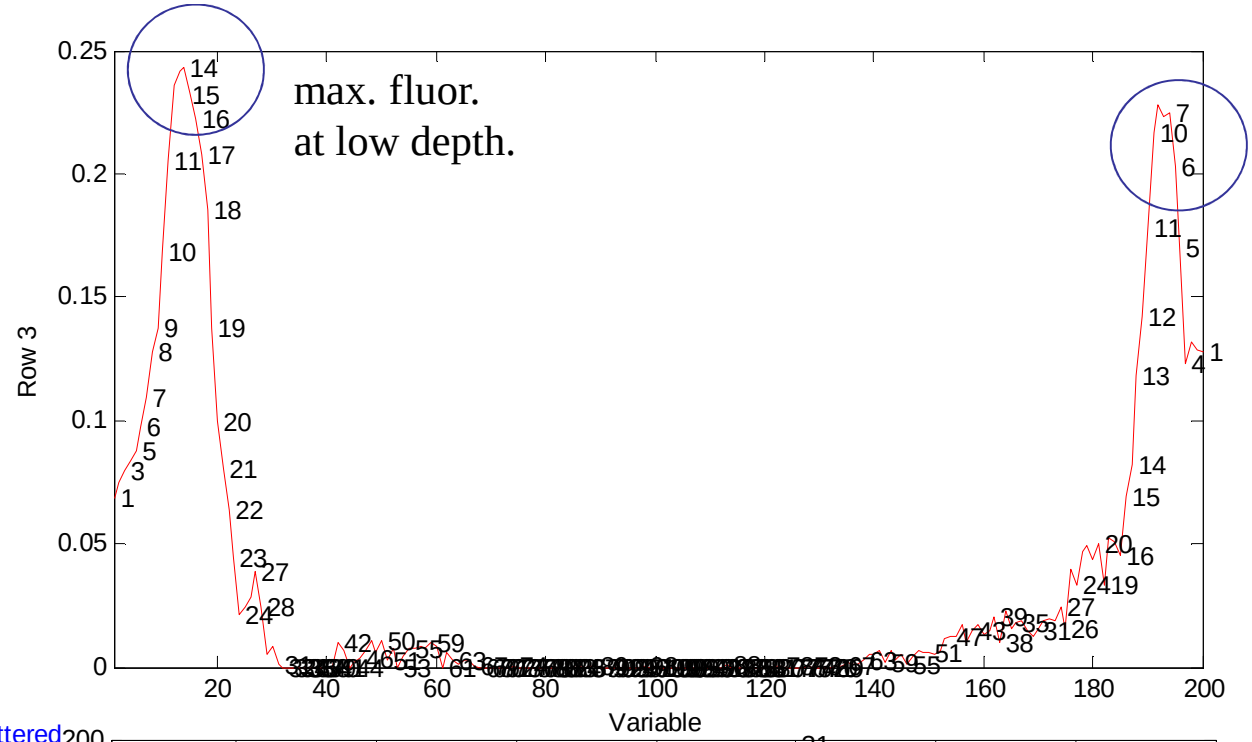


MCR-ALS resolution of 4 fluorescence patterns

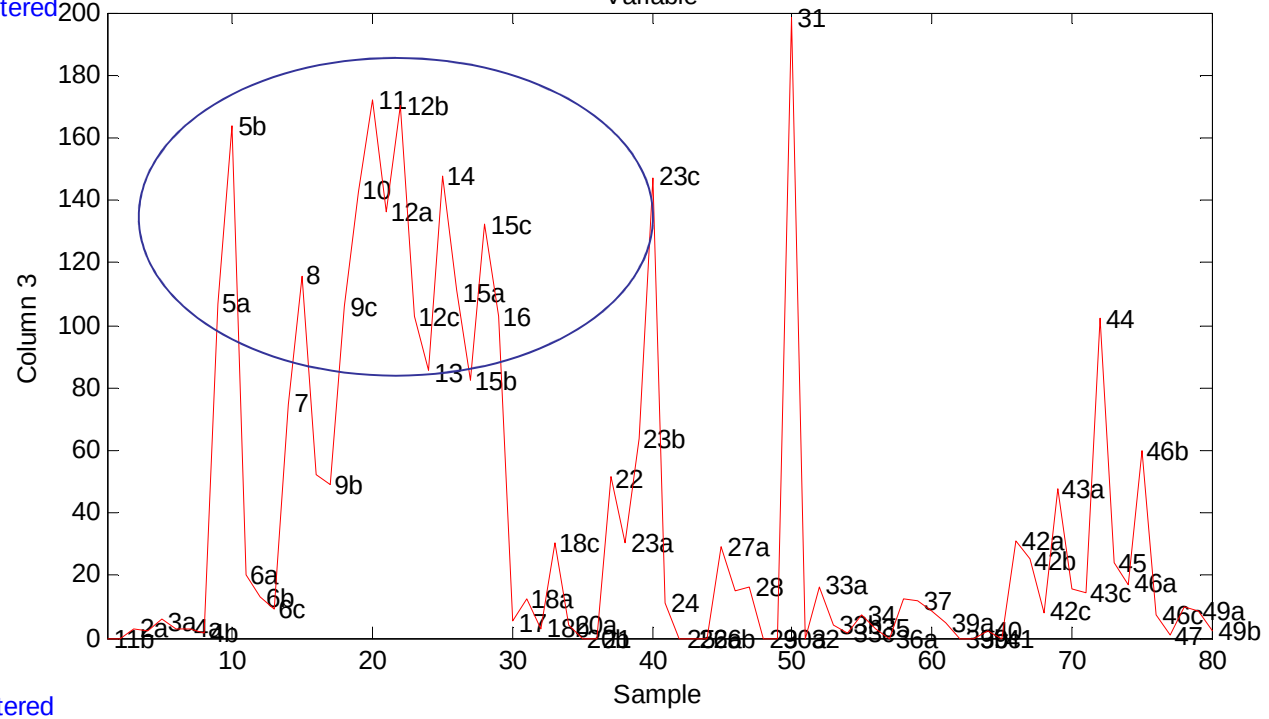


Decluttered

MCR-ALS C1

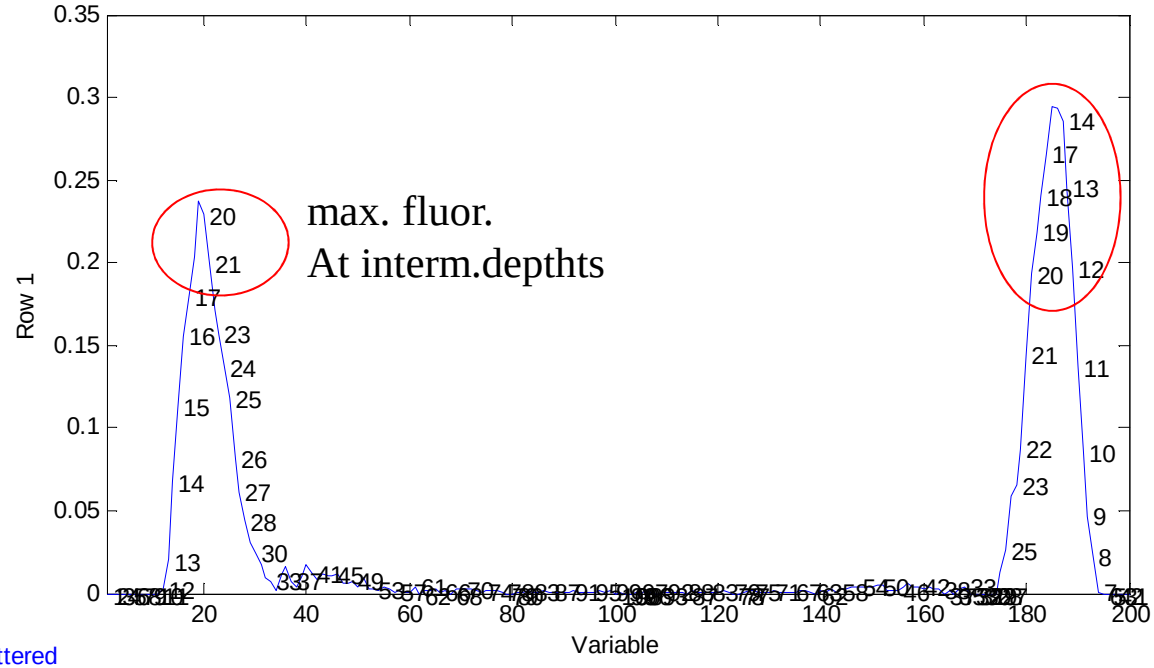


Decluttered

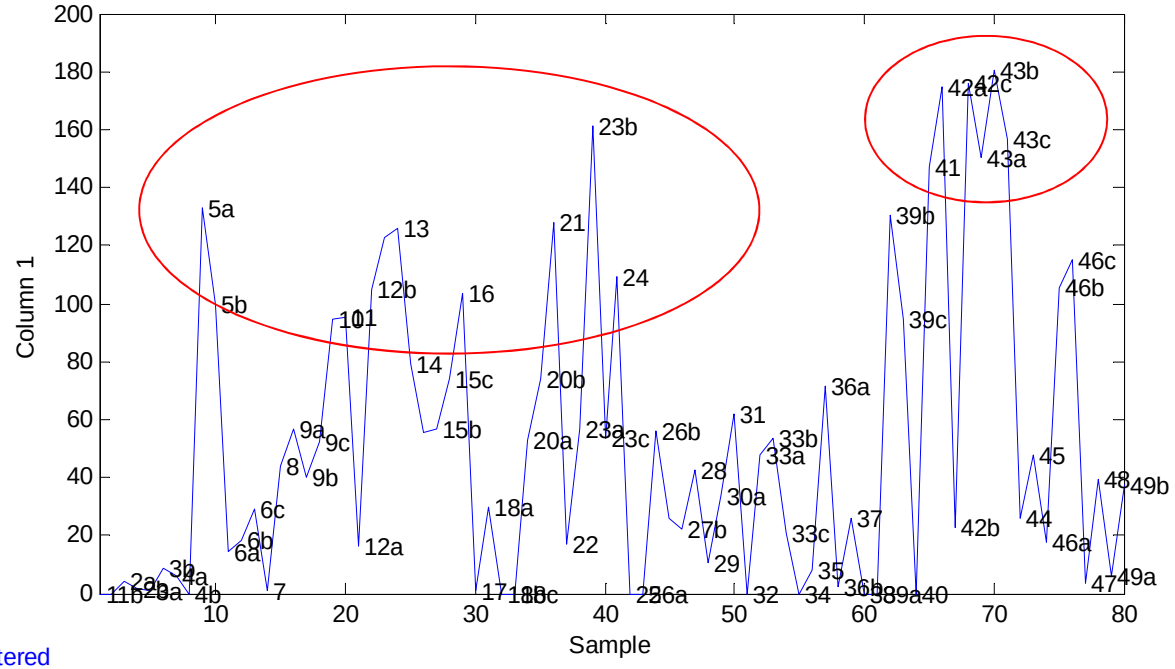


Decluttered

MCR-ALS
C1

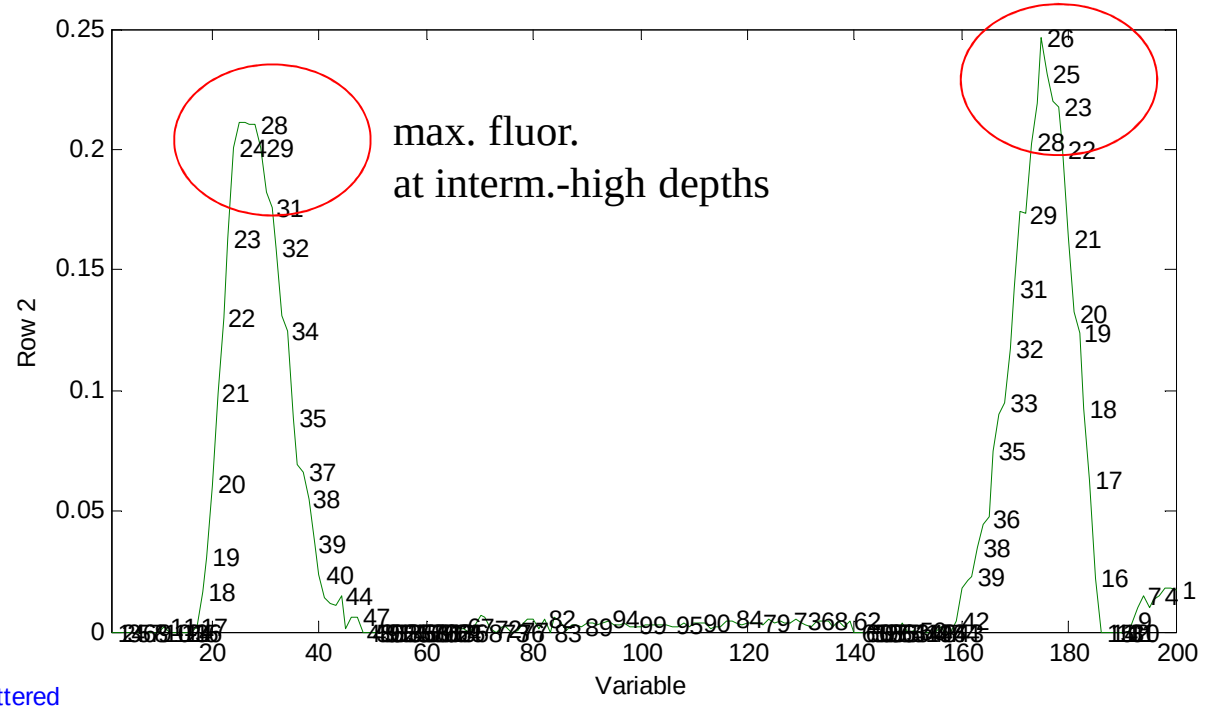


Decluttered

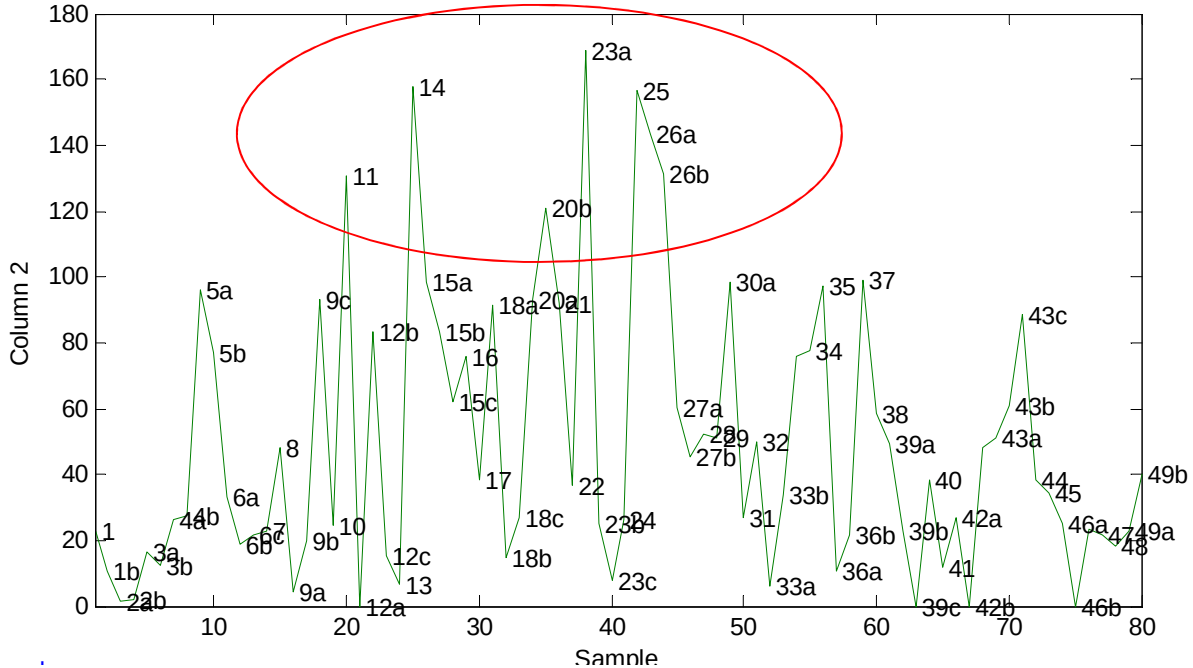


Decluttered

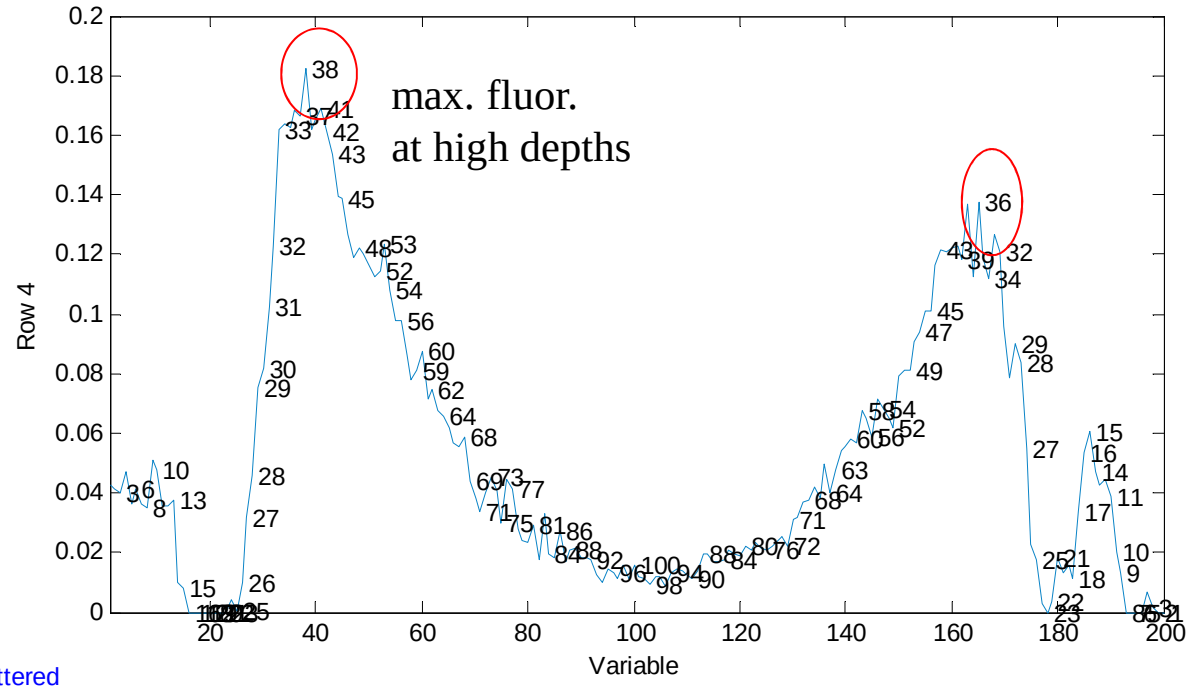
MCR-ALS
C3



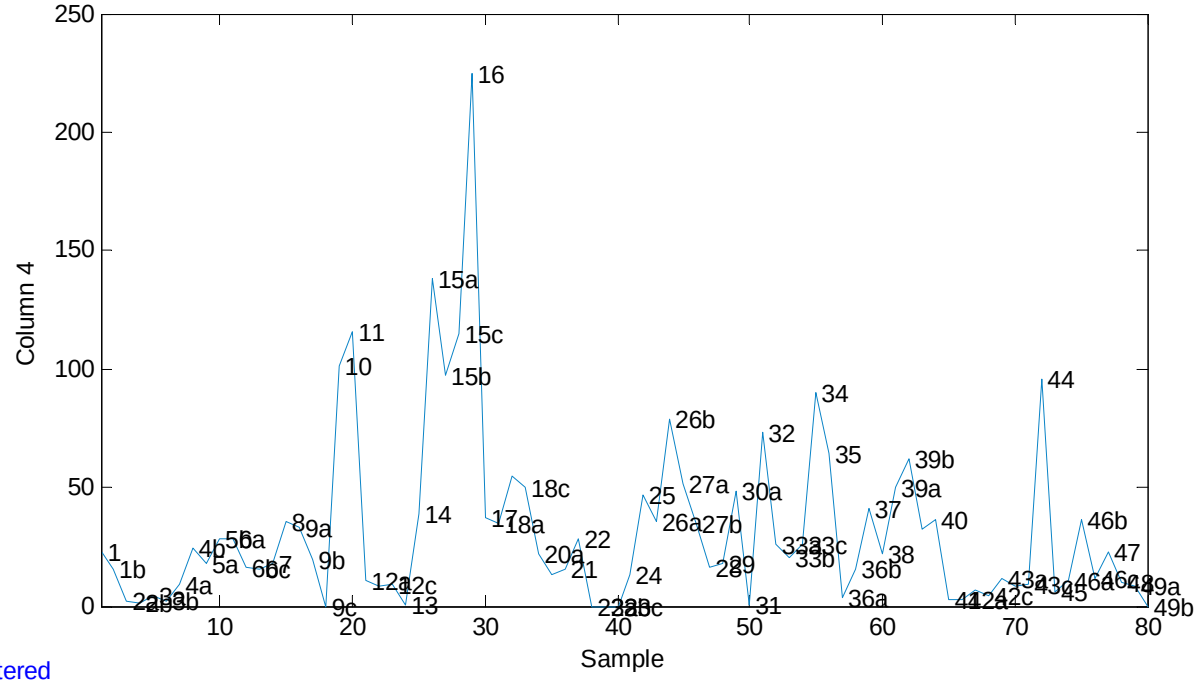
Decluttered



MCR-ALS
C4



Decluttered

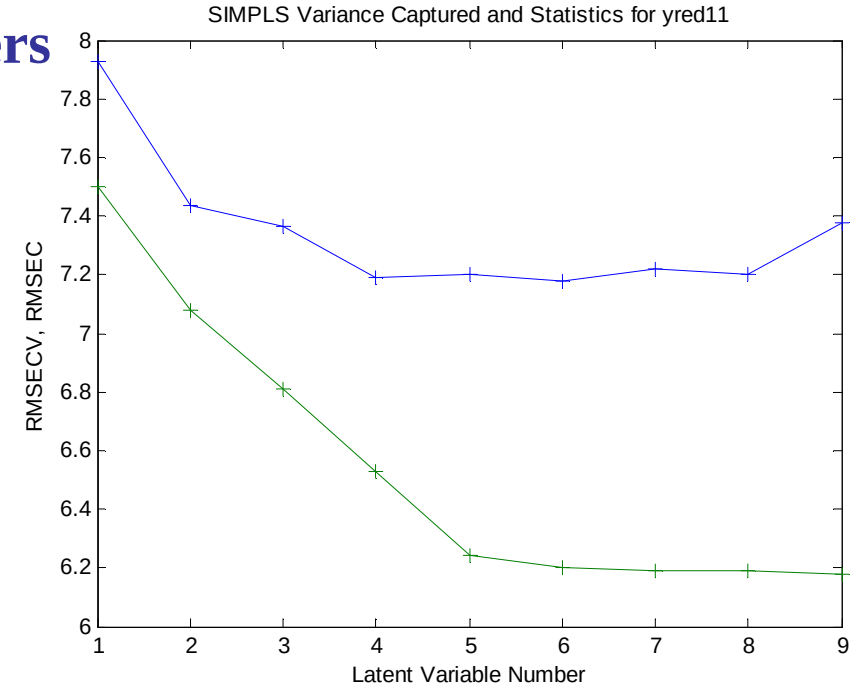


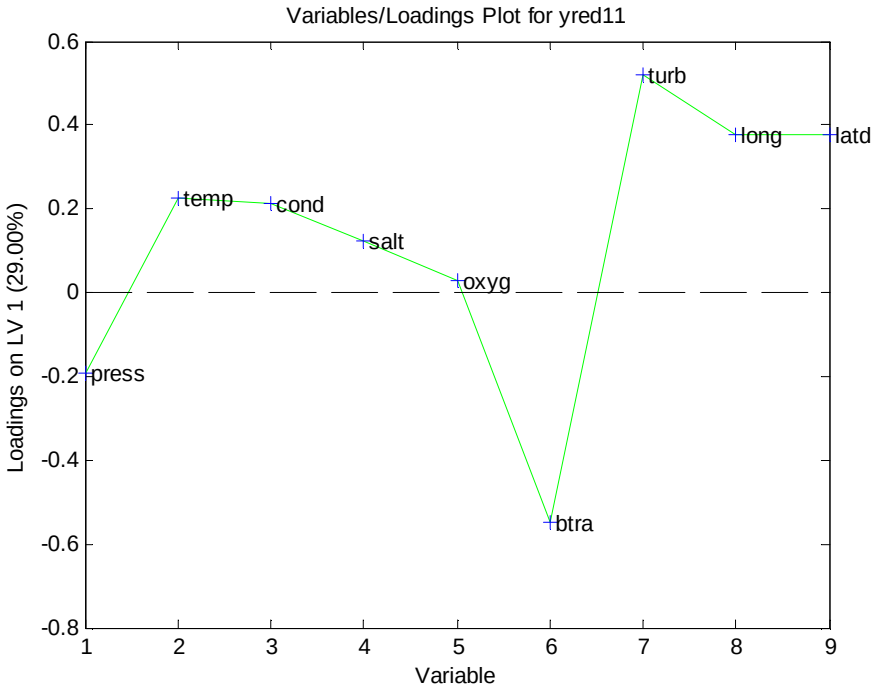
Decluttered

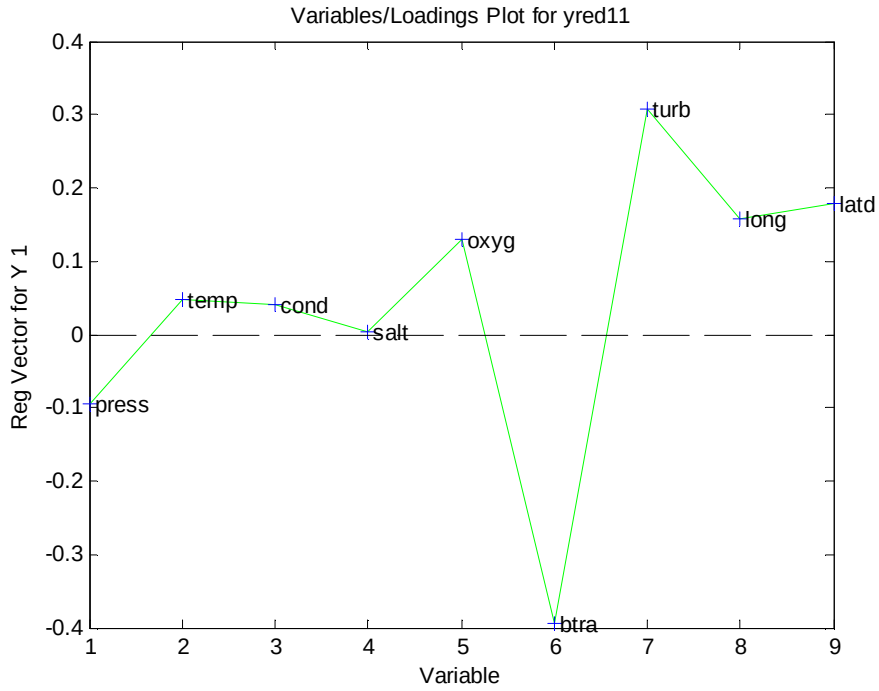
Fluorescence PLS prediction from others

Linear regression model using
Partial Least Squares calculated with SIMPLS
Cross validation: random samples w/ 8 splits
Percent Variance Captured by Regression Model

	-----X-Block-----		-----Y-Block-----	
Comp	This	Total	This	Total
1	29.00	29.00	76.95	76.95
2	**38.72**	**67.72**	**2.51**	**79.46**



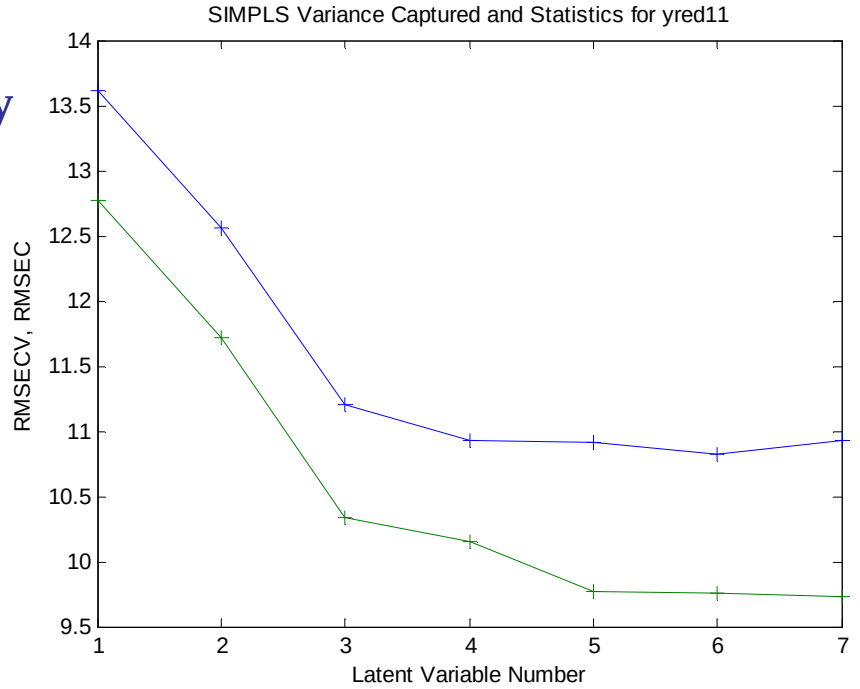


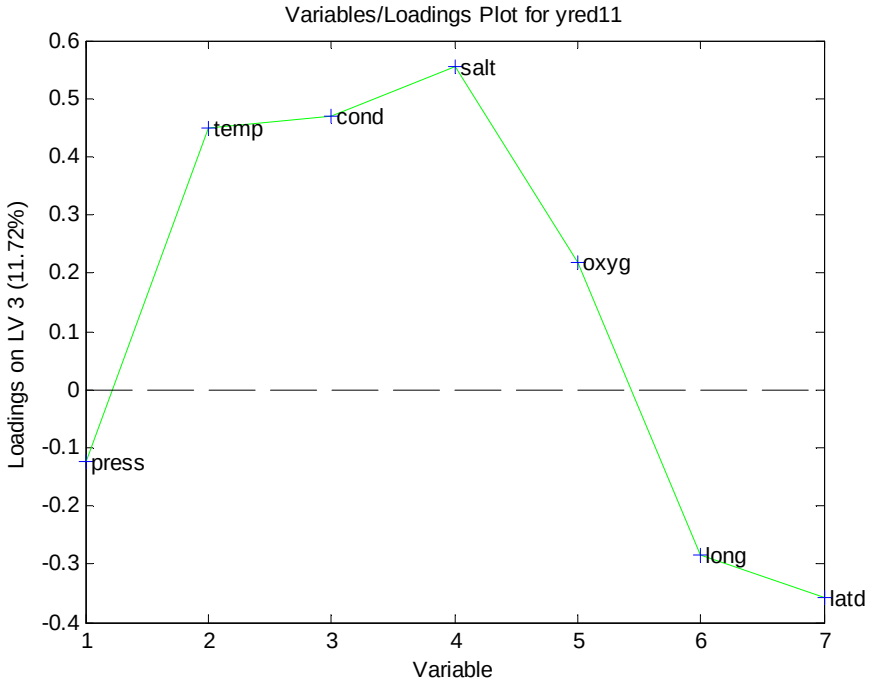


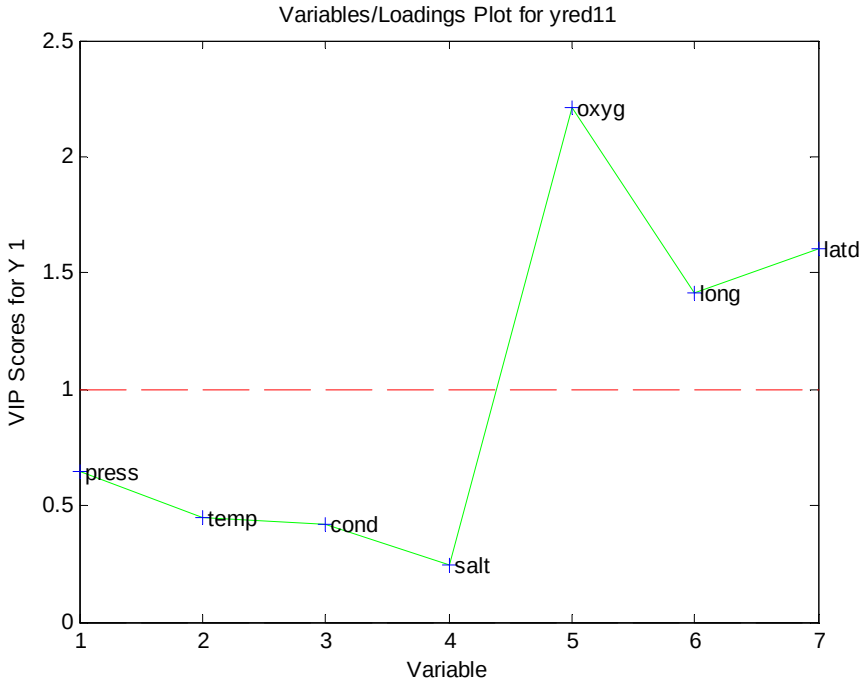
Fluorescence PLS prediction from others
Excluding beam transmission and turbidity

Linear regression model using
Partial Least Squares calculated with the SIMPLS
Cross validation: random samples w/ 8 splits
Percent Variance Captured by Regression Model

	-----X-Block-----		-----Y-Block-----	
Comp	This	Total	This	Total
1	34.94	34.94	33.23	33.23
2	39.55	74.49	10.53	43.76
3	**11.72**	**86.21**	**12.44**	**56.20**





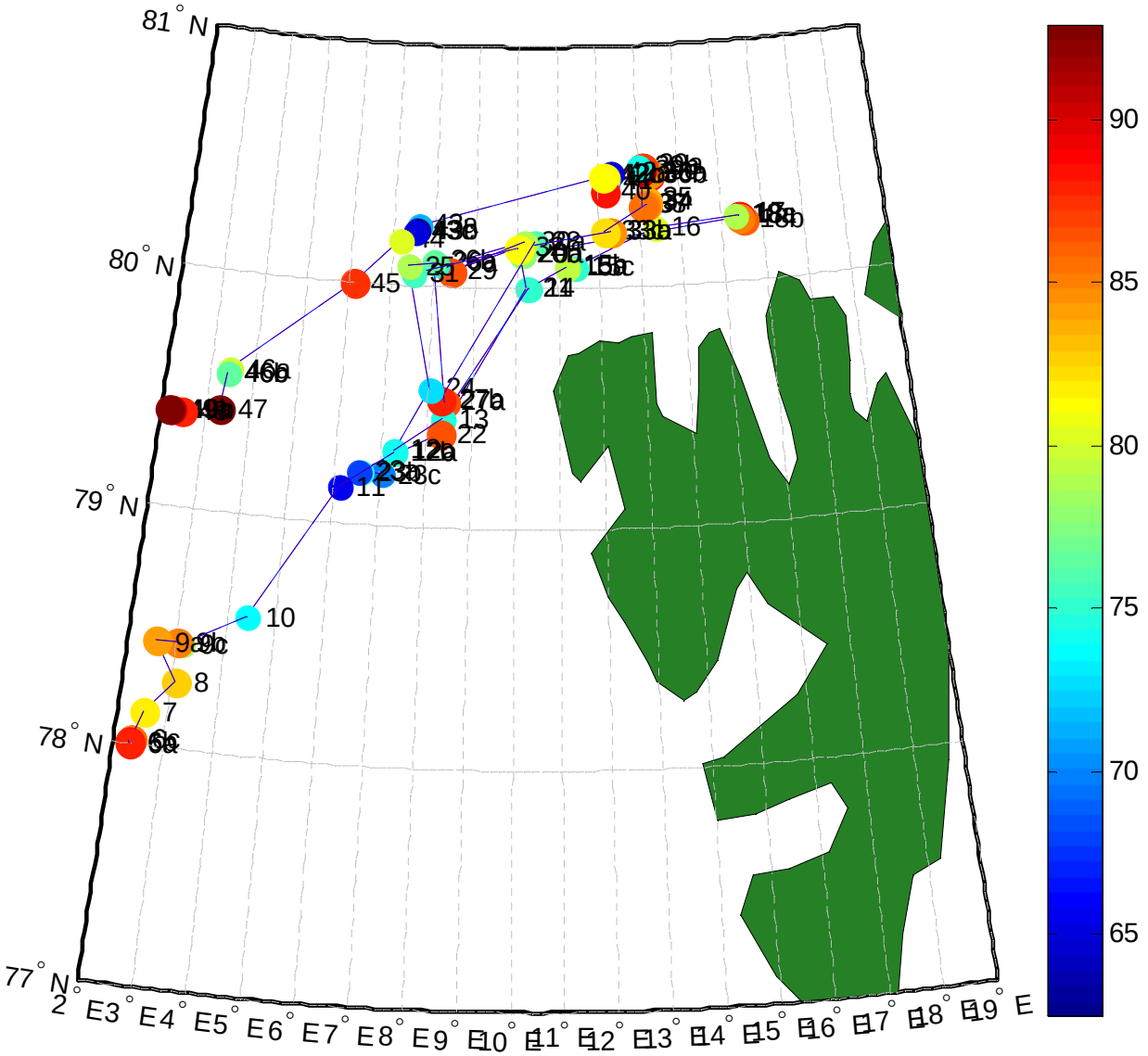


Representation of the fluorescence maxima

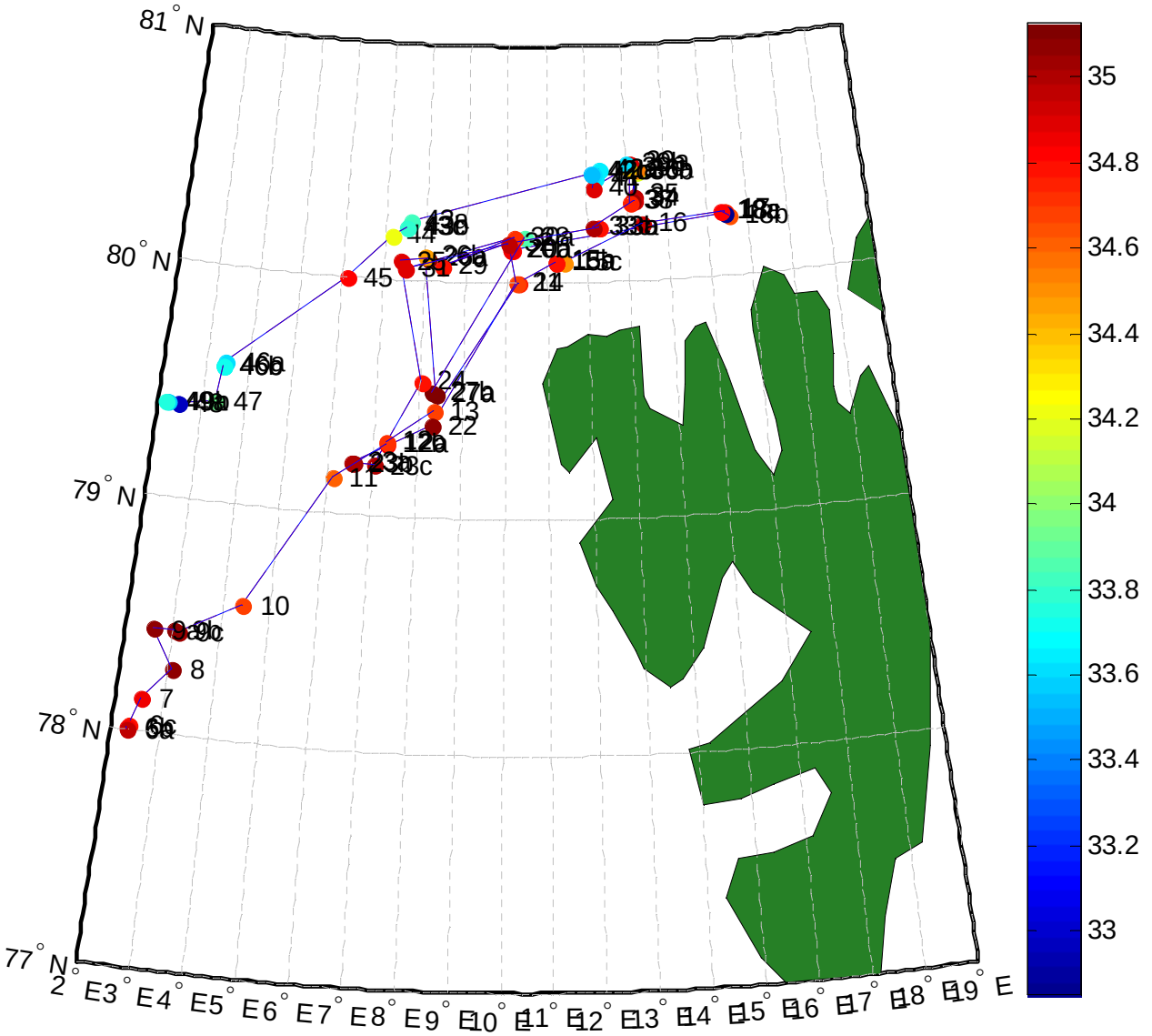
```
function mapatos(latlim,lonlim,latdr,longr,ident,y)
% function mapatos(latlim,lonlim,latdr,longr,ident,y)
%
mapatos([77,81],[2,19],latdr2(9:80,:),longr2(9:80,:),identc3(9:80,:),yred(9:80,6));

% worldmap of the desired lat and long, with the coast lines
worldmap(latlim,lonlim)
load coast
geoshow('landareas.shp', 'FaceColor', [0.15 0.5 0.15])
plotm(latdr,longr,'r--')
textm(latdr,longr+0.4,ident)
scatterm(latdr,longr,y,y,'filled');
gridm on

linem(latdr,longr,y);
```

10 CTD measured variables:
1 Depth, press
2 Temperature, temp
3 Conductivity, cond
4 Salt concentration, salt
5 Oxygent dissolved, oxyg
6 beam light transmission, btrm
7 fluorescence, fluor
8 turbidimetry, turb
9 latitude, latd
10 longitude, long

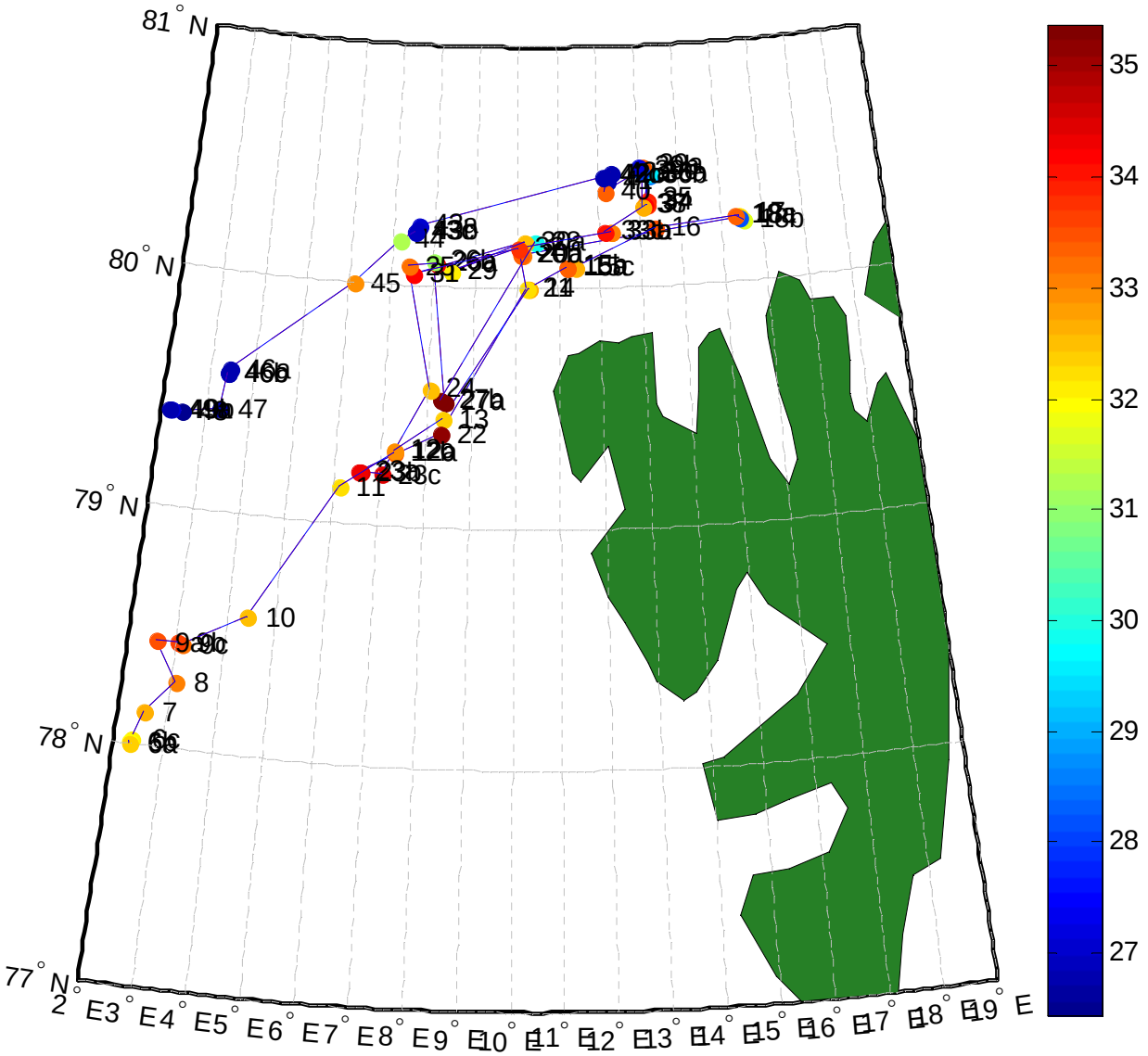
Fluoresc in dcm



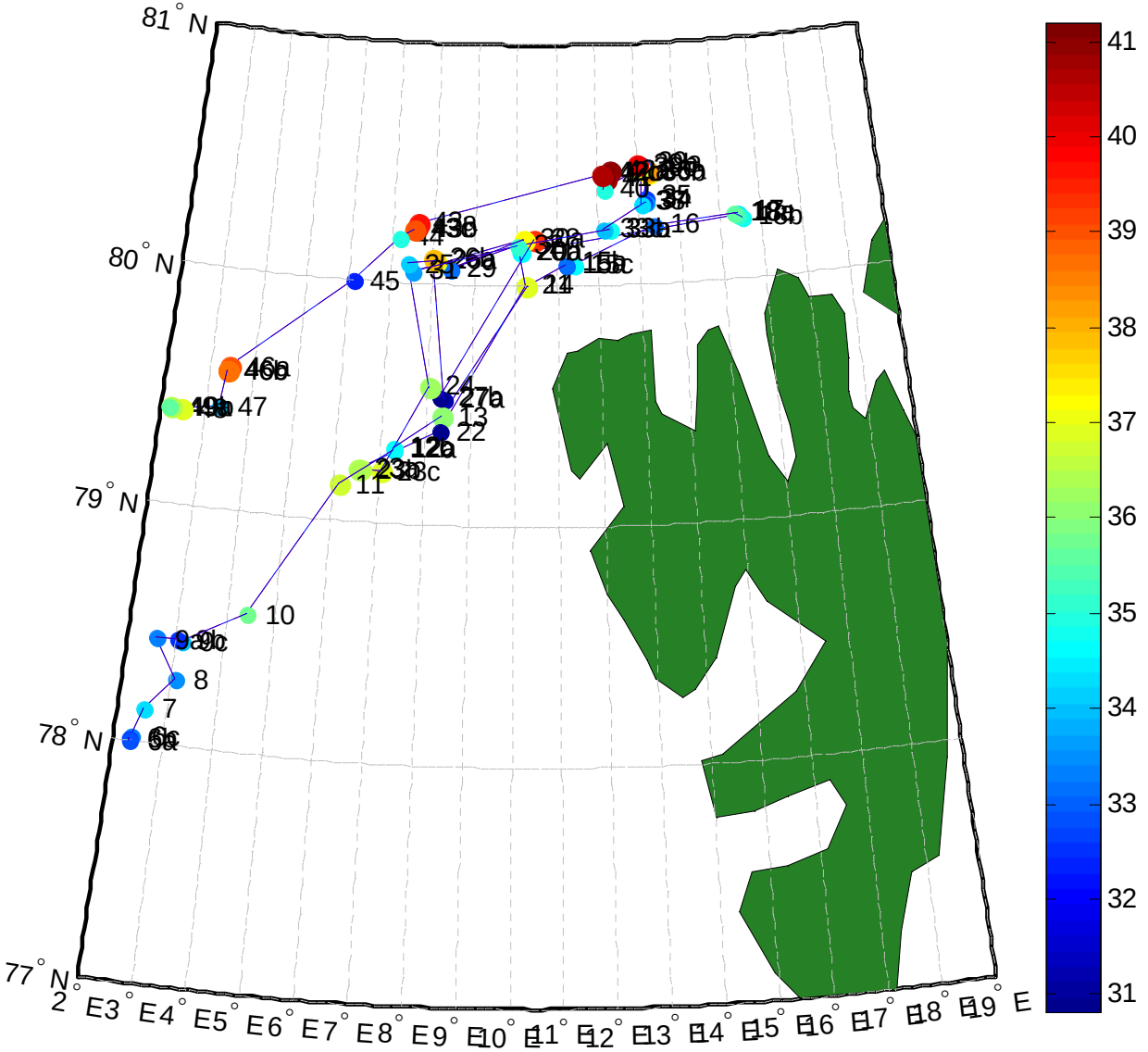
Salt conc in dcm



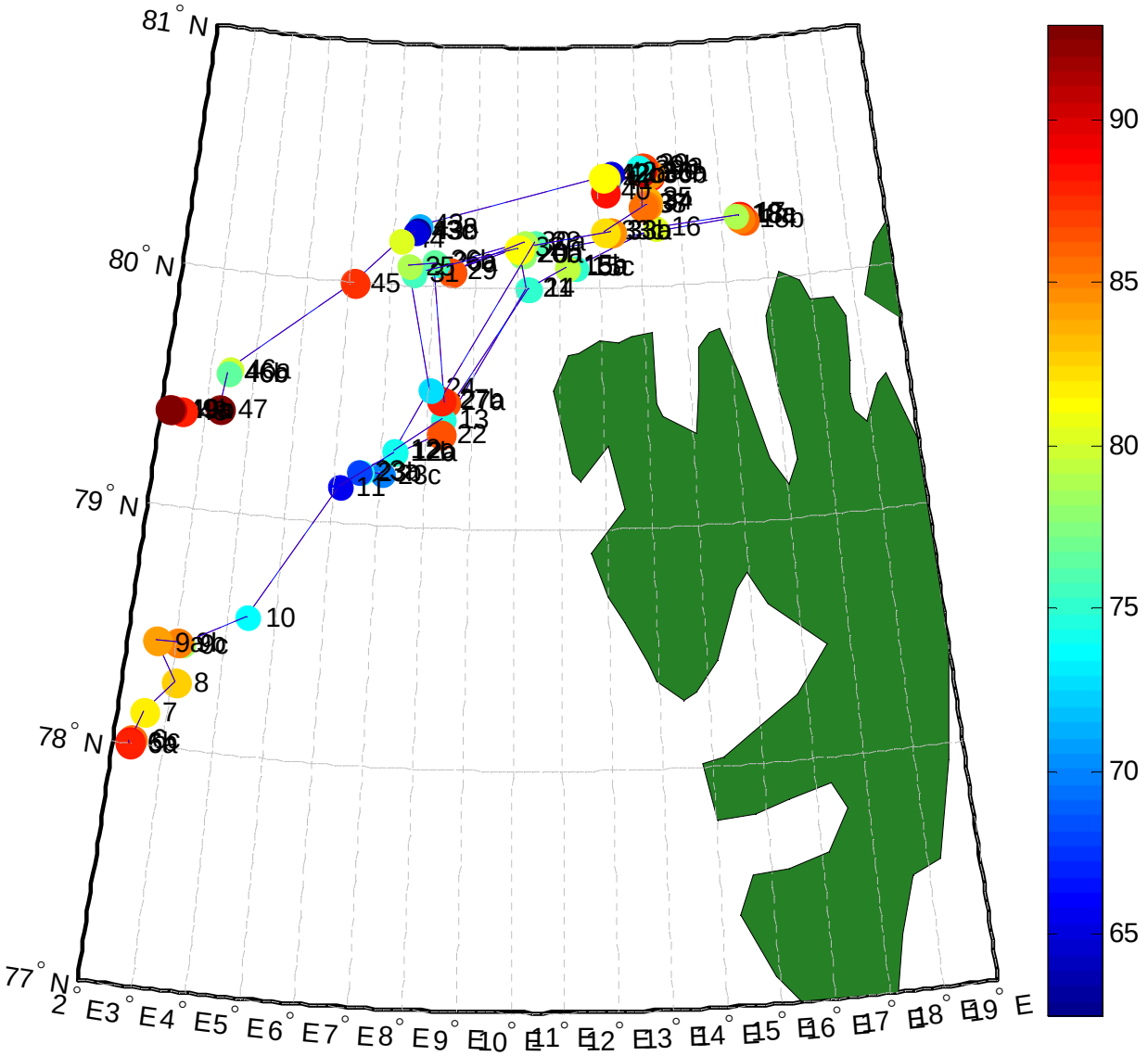
conduct in dcm



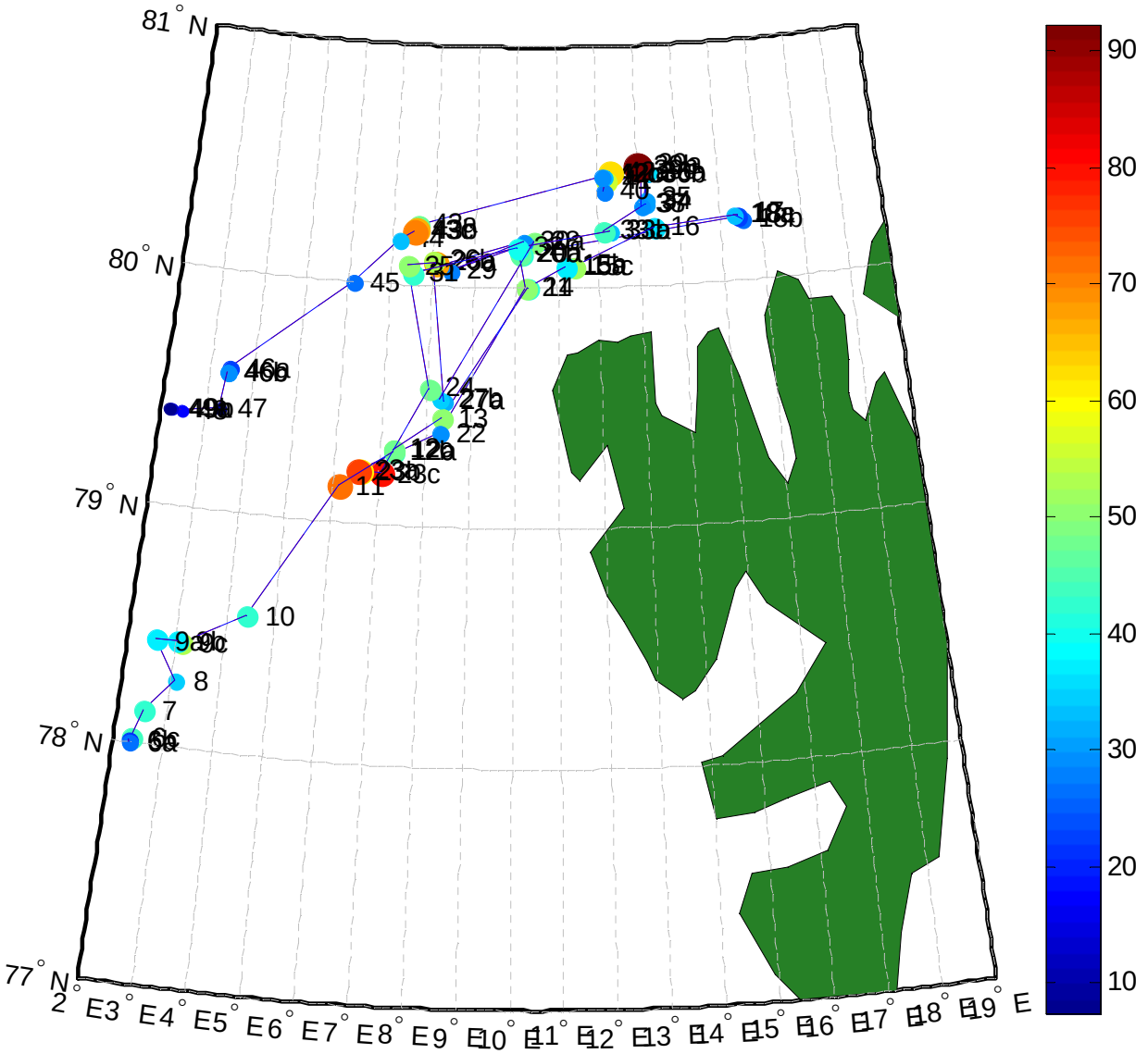
diss oxyg in dcm



beam transm in dcm



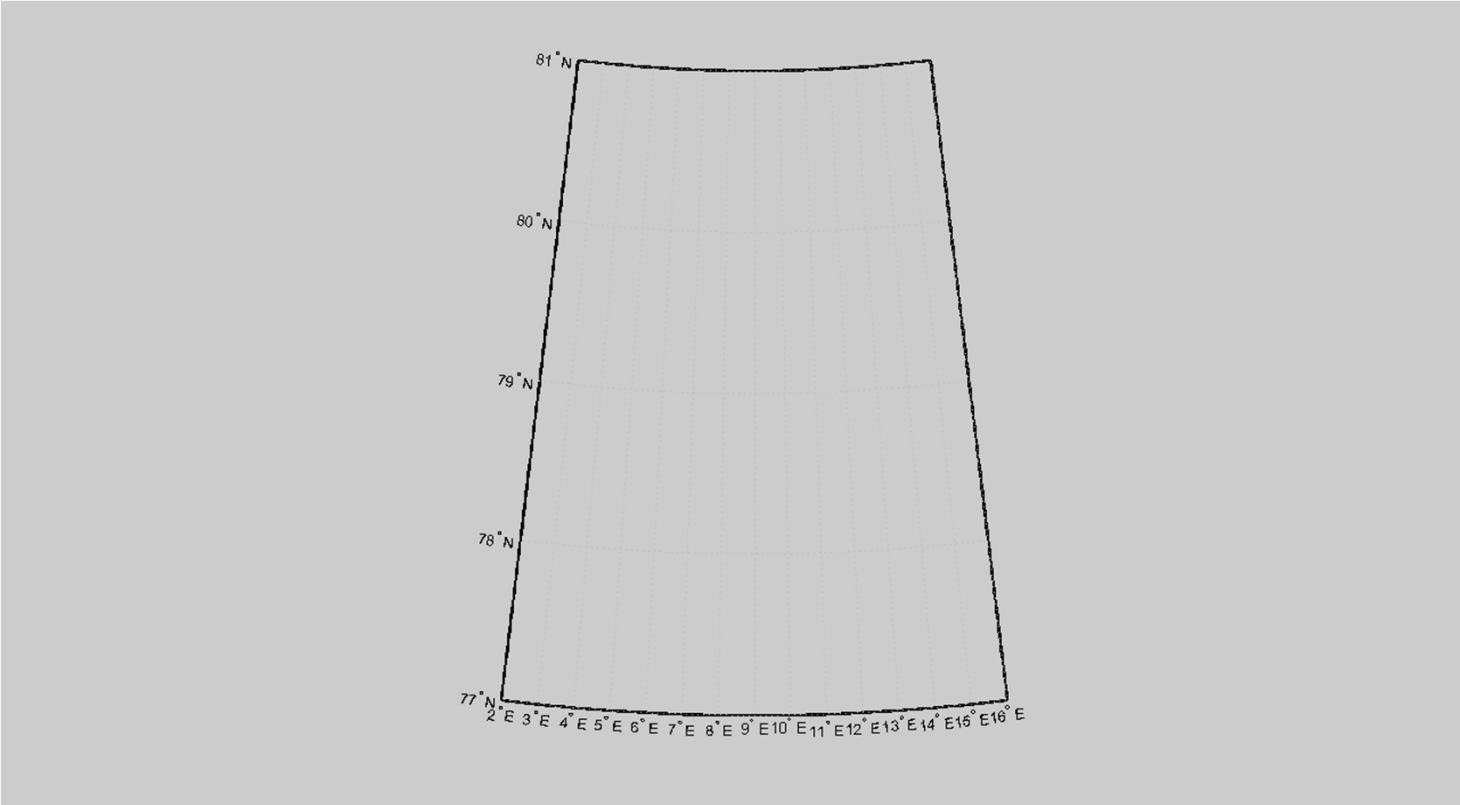
turbidimertry in dcm



From vectors to raster (grids)

[Z, refvec] = geoloc2grid(latdr2(9:80,:),longr2(9:80,:),yred(9:80,7),cellsize);

load atosmaps
[Z, refvec] = geoloc2grid(latdr2(9:80,:),longr2(9:80,:),yred(9:80,7),0.1);
worldmap([77,81],[2,16])

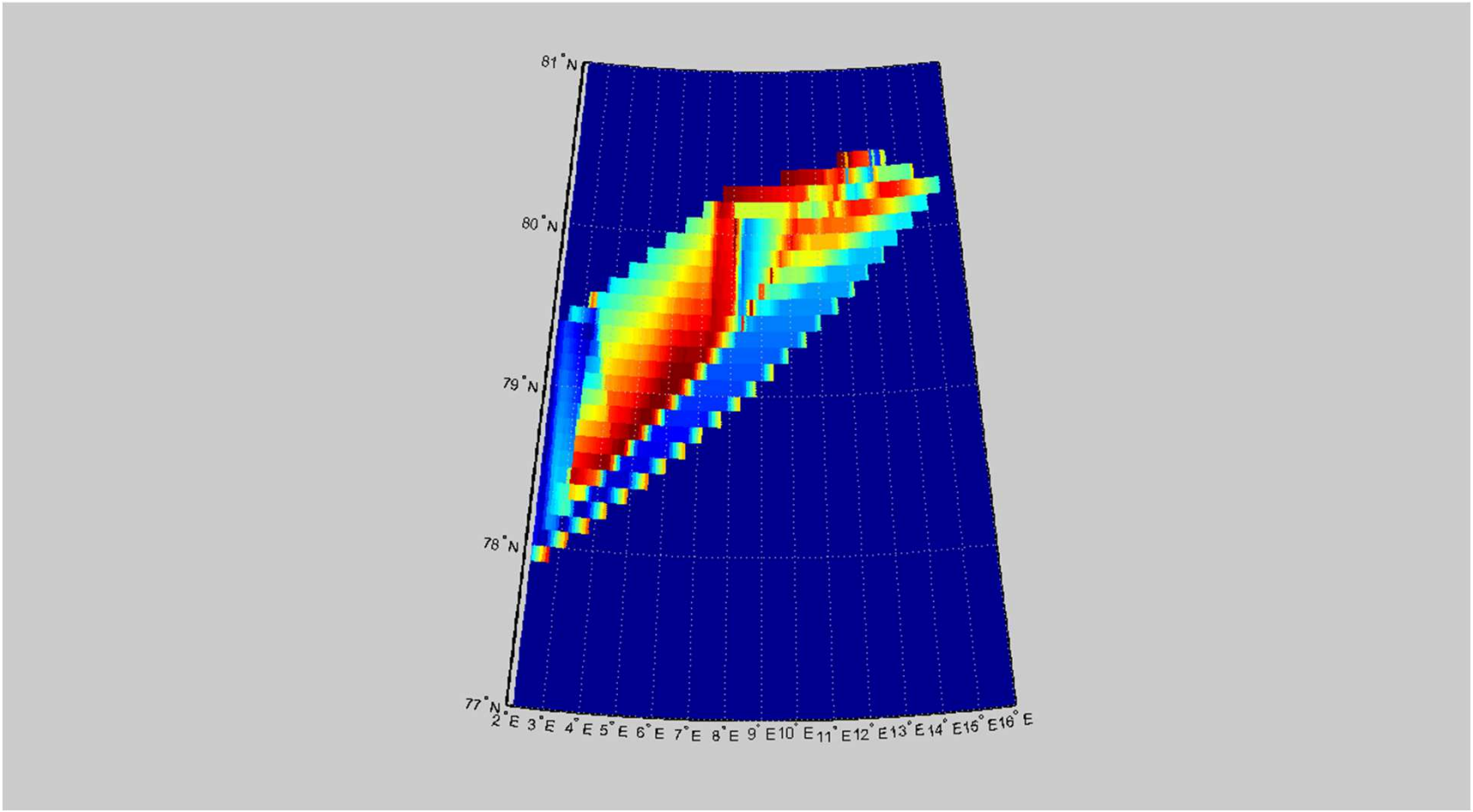


From vectors to raster (grids)

```
function [Z,refvec]=mapgridatos(latlim,longlim,latw,longw,y,cellsize)
% function [Z,refvec]=mapgridatos(latlim,longlim,lat,long,y,cellsize)
close
[Z, refvec] = geoloc2grid(latw,longw,y,cellsize);
worldmap(latlim,longlim)
meshm(Z,refvec)
load coast
geoshow('landareas.shp', 'FaceColor', [0.15 0.5 0.15])
plotm(latw,longw,'k+','LineWidth',1)
linem([79.5,80.5],[2,18],'w--','LineWidth',3)
colorbar
```

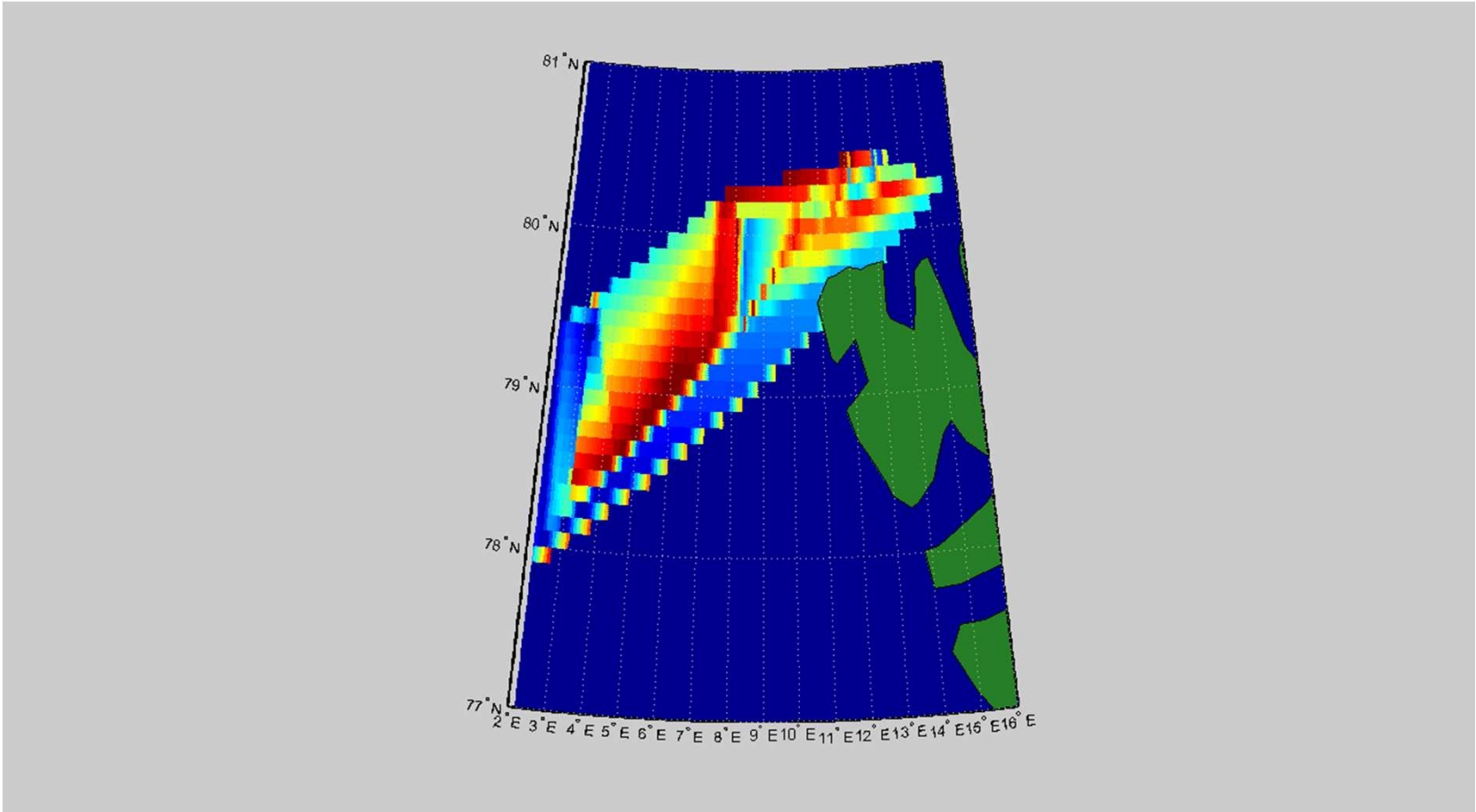
From vectors to raster (grids)

meshm(Z,refvec)



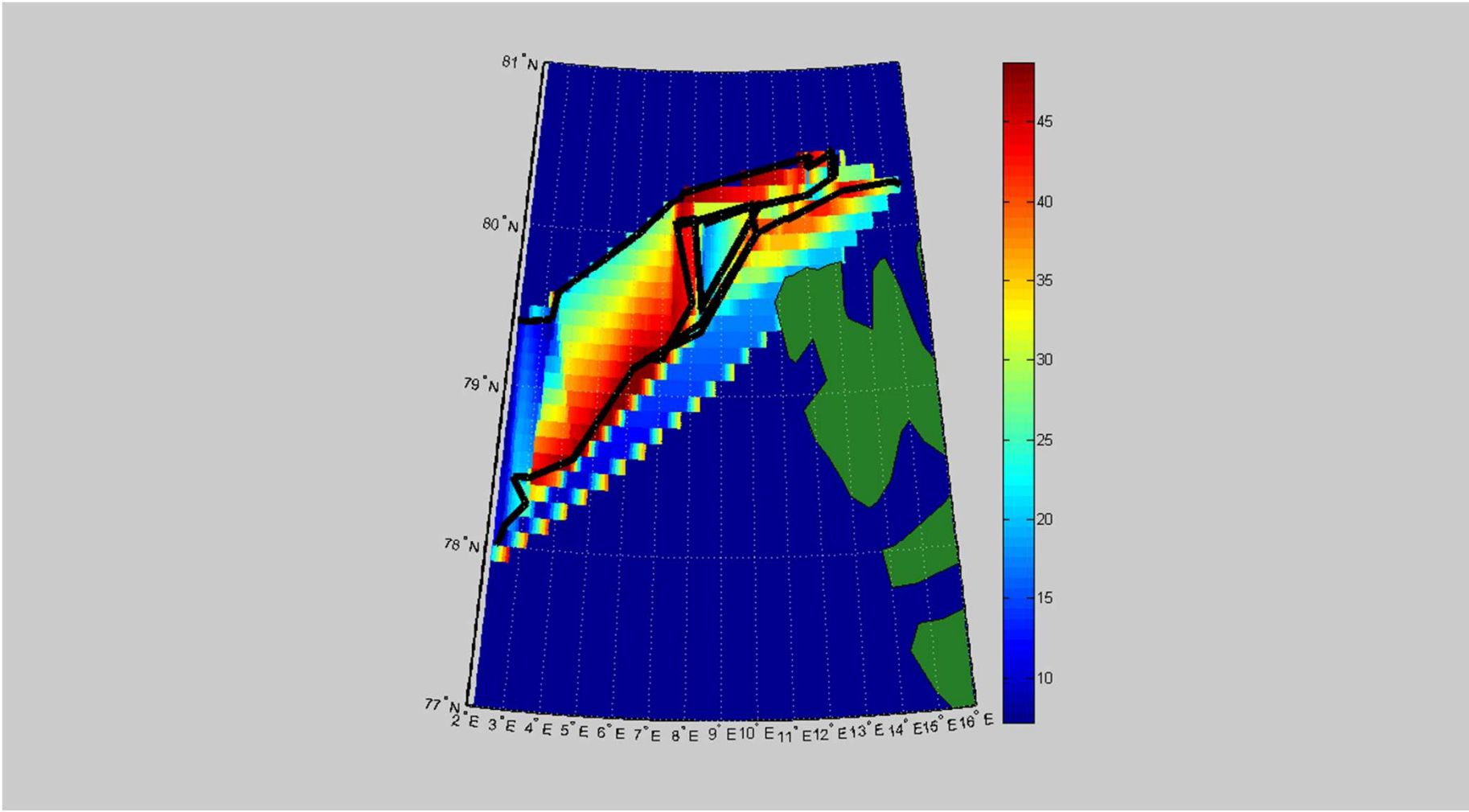
From vectors to raster (grids)

```
>> load coast
>> geoshow('landareas.shp', 'FaceColor', [0.15 0.5 0.15])
```



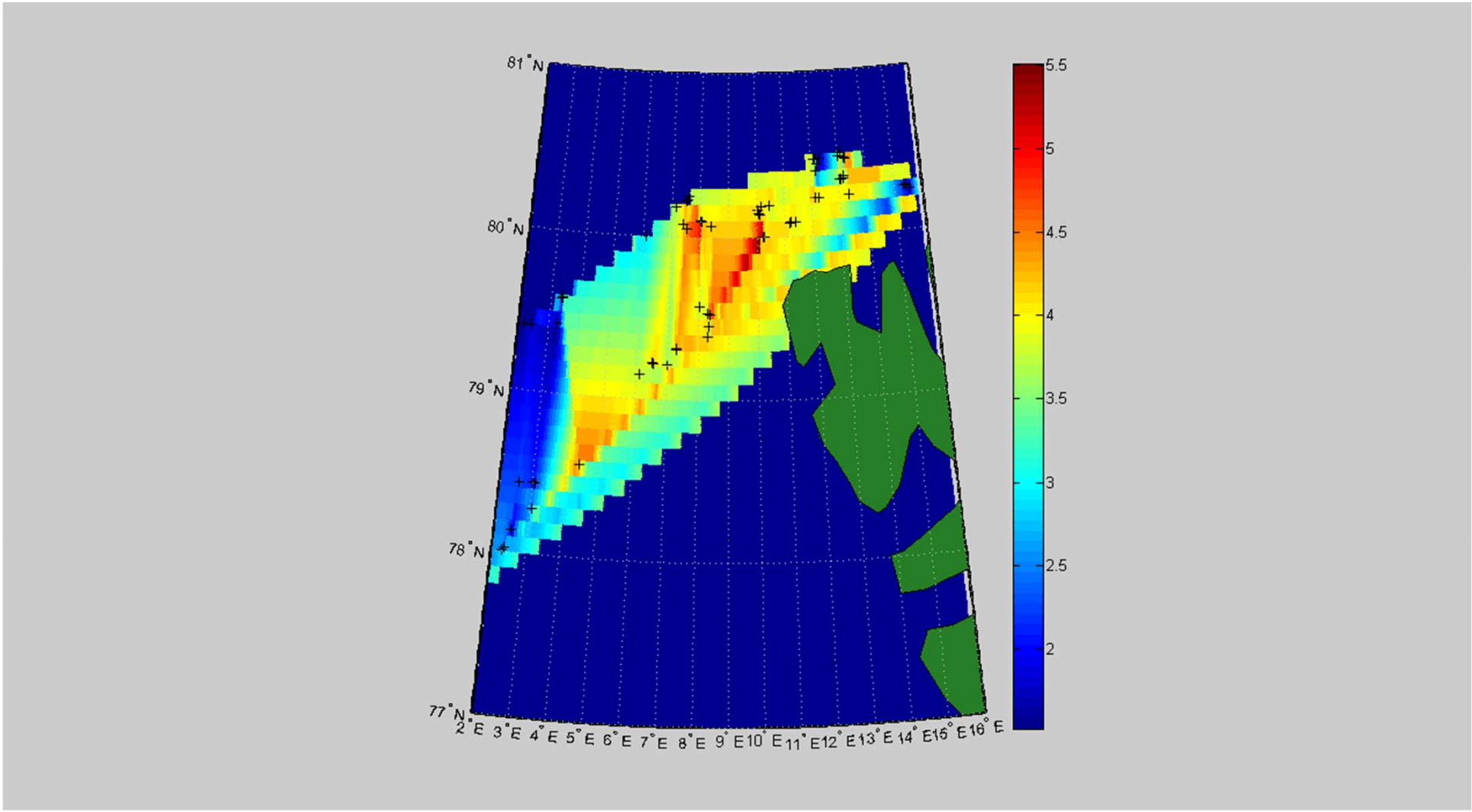
From vectors to raster (grids)

plotm(latdr2(9:80,:),longr2(9:80,:),'k-','LineWidth',5)
colorbar



From vectors to raster (grids)

[Z,refvec]=mapgridatos([77,81],[2,16],latdr(9:80,:),longr(9:80,:),ydeep(9:80,2),0.1);



Investigation of Arctic and Antarctica spatial and depth patterns of sea water in CTD profiles using chemometric data analysis.

Ewelina Kotwa, Silvia Lacorte, Carlos Duarte, Roma Tauler

Polar Science, in press

In this paper a comparative study of CTD (Conductivity-Temperature-Depth) sensor using 2- and 3-way chemometric methods was performed for exploratory analysis and spatial patterns discovery from two distinct polar geographical areas: Arctic and Antarctic Seas. Results from the two oceanographic expeditions taking place in July 2007 (Arctic) and February 2009 (Antarctica), spanning the areas of 68°N-81°N, 20°W-20°E and 60°S-70°S, 50°W-77°50´W respectively, will be presented

CTD data is obtained by standard sensor devices frequently used in oceanographic research type of studies. These data sets can be arranged in 3-way data structures (according to sea water depth, measured variables and geographical sample location modes). Measured variables were collected within the total number of 174 locations. Special importance was given to correlation and possible prediction of fluorescence (mostly related to biological activity), from other physical variables.

Data pre-processing together with outliers and missing values detection and elimination strategies were applied before the actual exploratory data analysis was performed. Classical and robust versions of common chemometric tools such as PCA, PARAFAC, PLS and nPLS were applied for data exploration and calibration, and MATLAB's mapping toolbox was used for results geo-referencing and visualization.

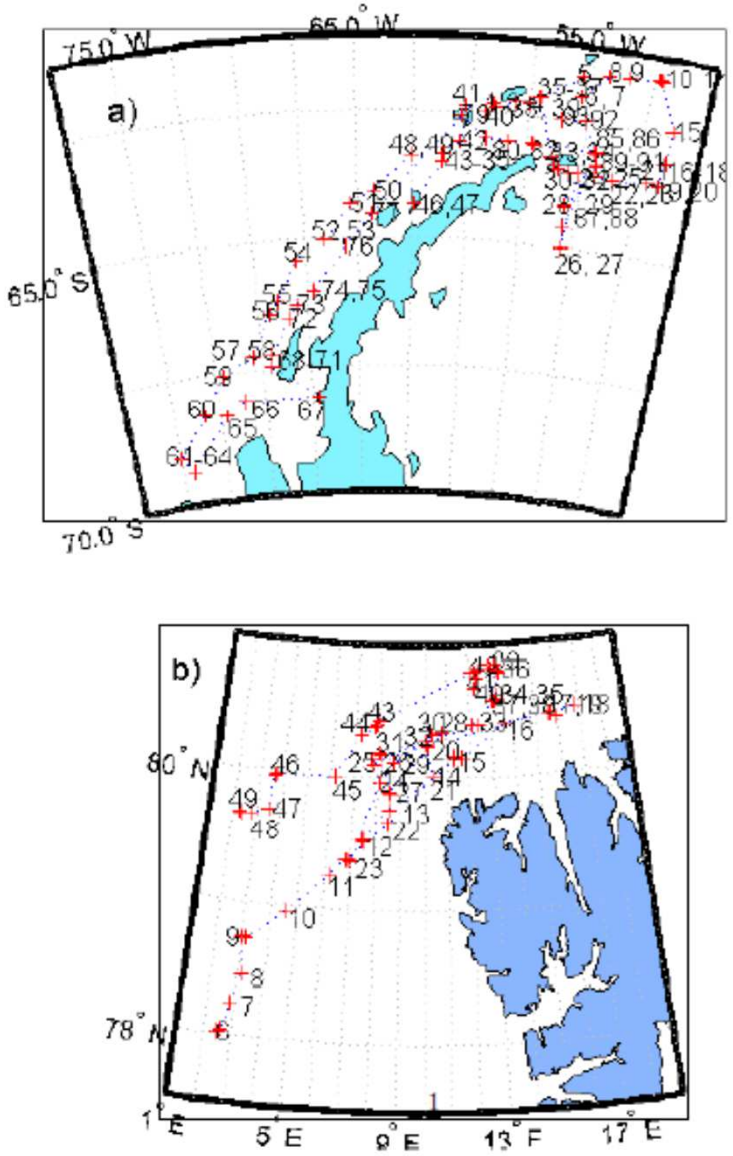


Figure 1: Locations of CTD measurement stations in the Antarctica (a) and in the Arctic Sea (b).

Measured variables.

Table 1: Variables measured at every station in both, the Arctic Sea and the Antarctica.

No.	Variable	Unit
1	Temperature	[ITS-68, deg C]
2	Conductivity	[mS/cm]
3	Salinity	[PSU]
4	Oxygen	SBE 43 [ml/l]
5	Beam Transmission	[%]
6	Fluorescence	arbitrary units [AU]
7	Sea-point Turbidity	[FTU]

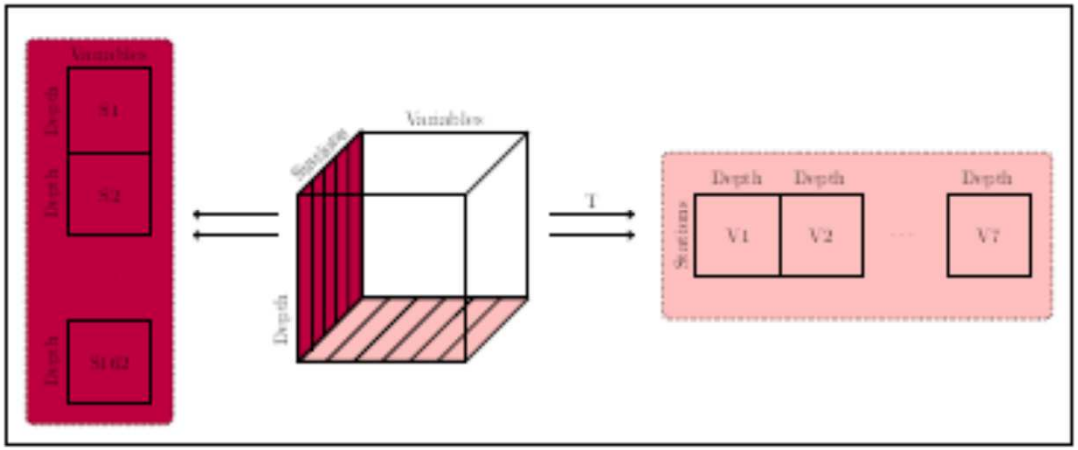


Figure 2. Arrangement of the CTD data in a cubic structure according to three modes: depth, variables and stations. Two unfolding directions, variable- and station-wise were adapted to fit the 2-way workframe.

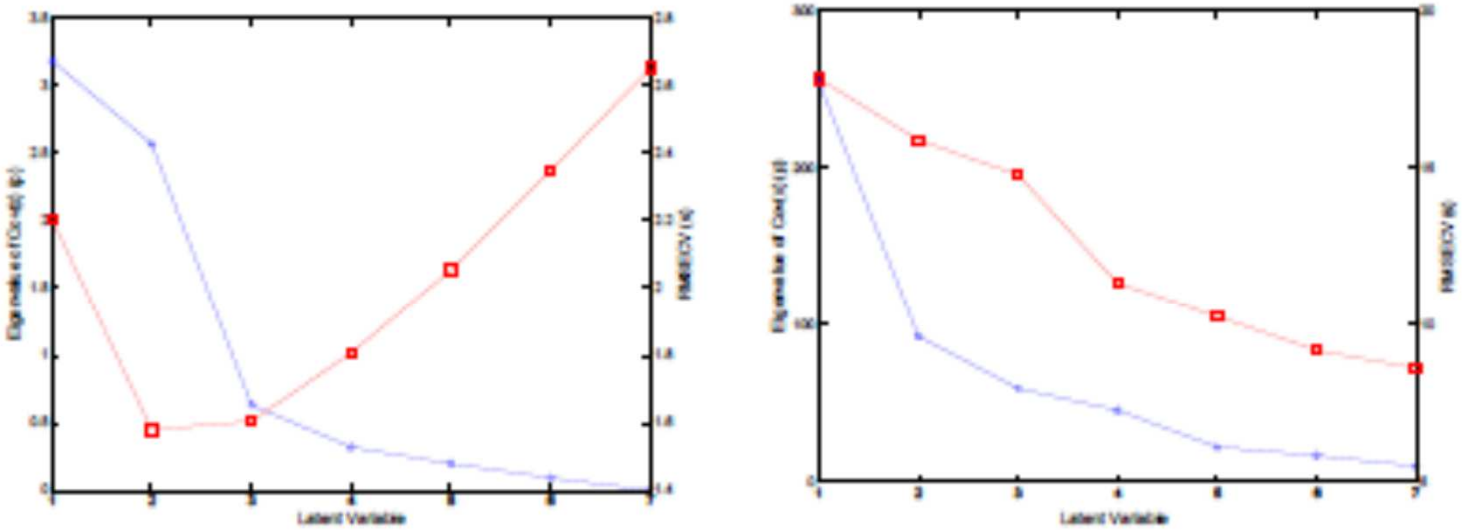


Figure 3. Cross-Validated Root Mean Square Error (red squares) and eigenvalues of the covariance matrix X for the PCA model, for X being a) variable-wise unfolded data; b) station-wise unfolded data.

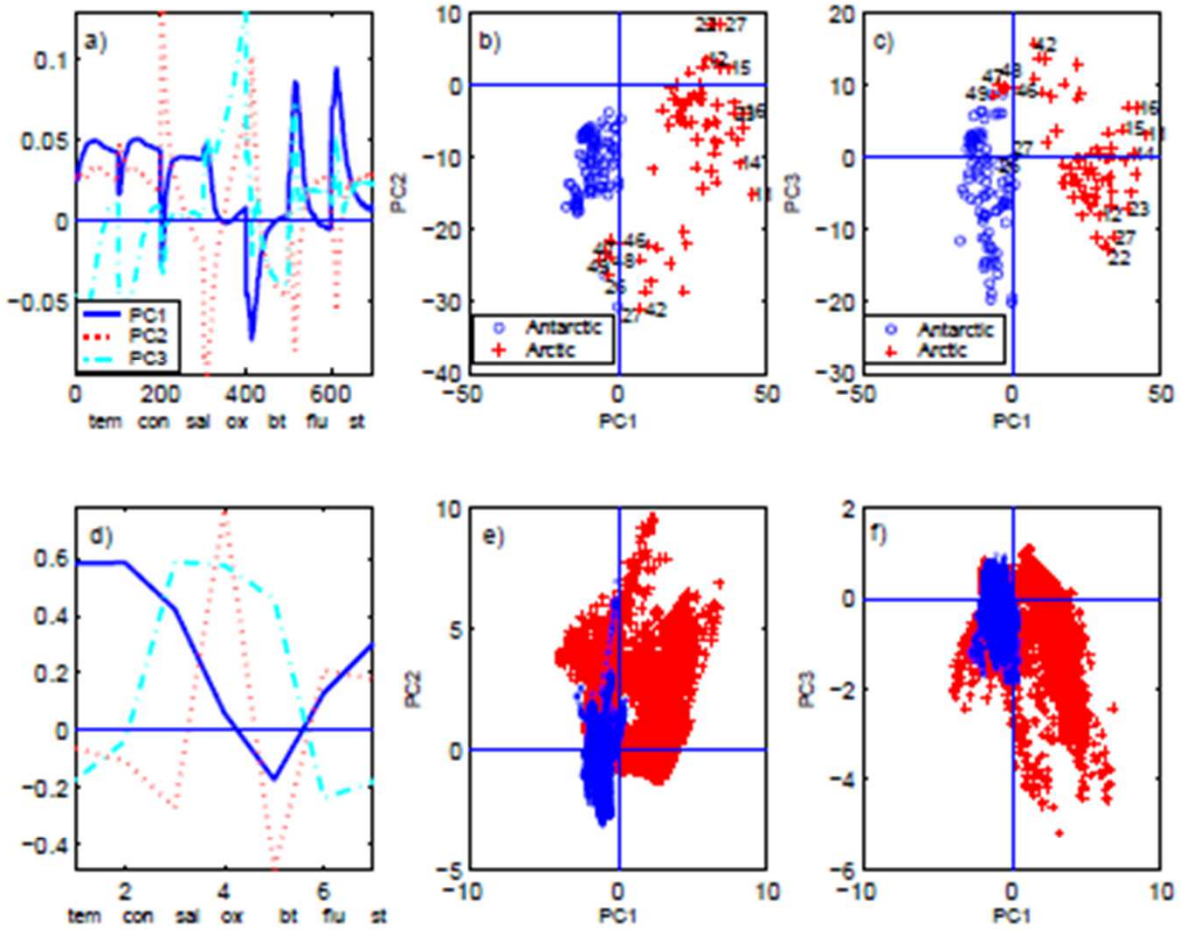


Figure 1: Loadings (panels a) and d)) and scores (panels b), c), e) and f)) for PCA models on station- (up) and variable-wise (down) unfolded data.

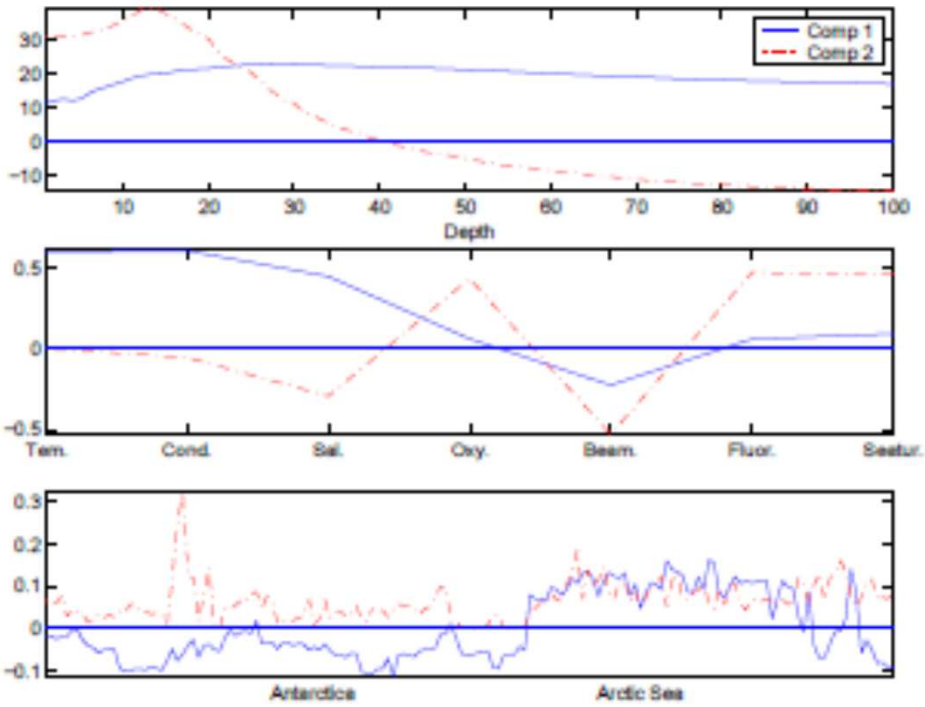
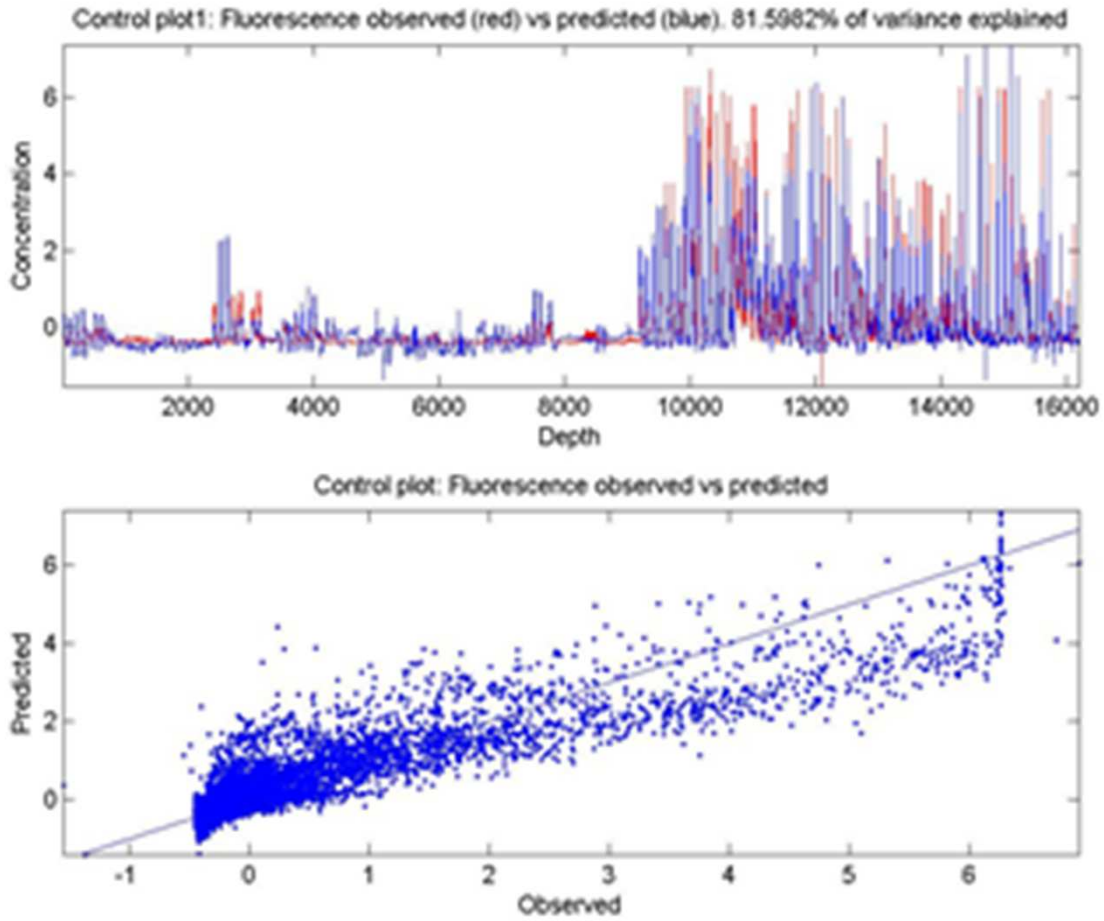


Figure 5. Resolved profiles by 2-component PARAFAC model according to a) depth, b) variable, c) station mode.

Table 2. Choosing amount of components in PARAFAC model according to four indices: explainced variance, core consistancy, number of iterations and total calculation time.

No.	Variance explained	Core consistancy	Iteration	Time
1	30.32	100	5	0.56
2	63.46	100	5	0.41
3	73.70	−481	24	1.06



Control plots for predicted Fluorescence values;

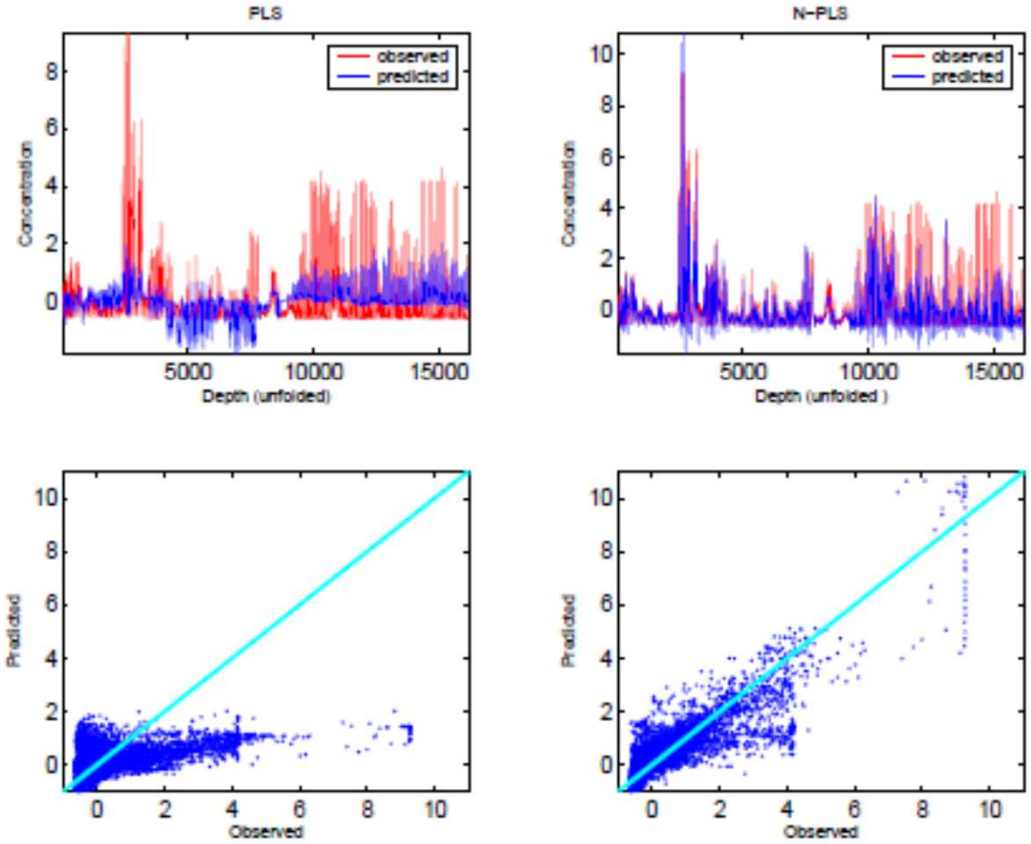


Figure 7. Predicted versus observed values for PLS (left) and N-PLS (right) models, with only physico-chemical variables entering X-block.

Table 3. Explained variance of PLS and N-PLS models for two varian of predictive block: 1. with all CTD variables; 2. with physico-chemic variables only.

		all variables		physico-chemical	
	No.	X block	Y block	X block	Y block
PLS	1	40.54	65.01	27.26	22.36
	2	70.30	70.47	81.42	22.64
	3	92.32	74.08	100	23.05
nPLS	1	26.66	60.40	46.00	31.56
	2	65.56	76.88	72.04	69.73
	3	72.55	85.50	83.42	79.13

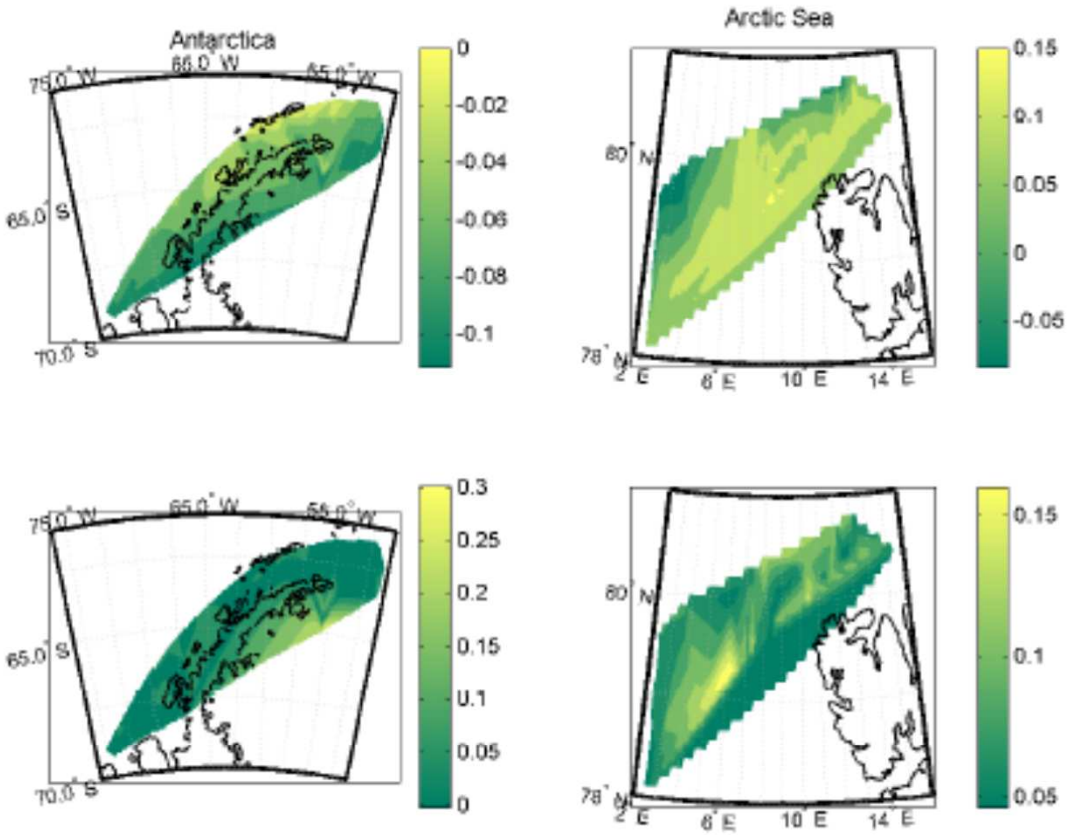


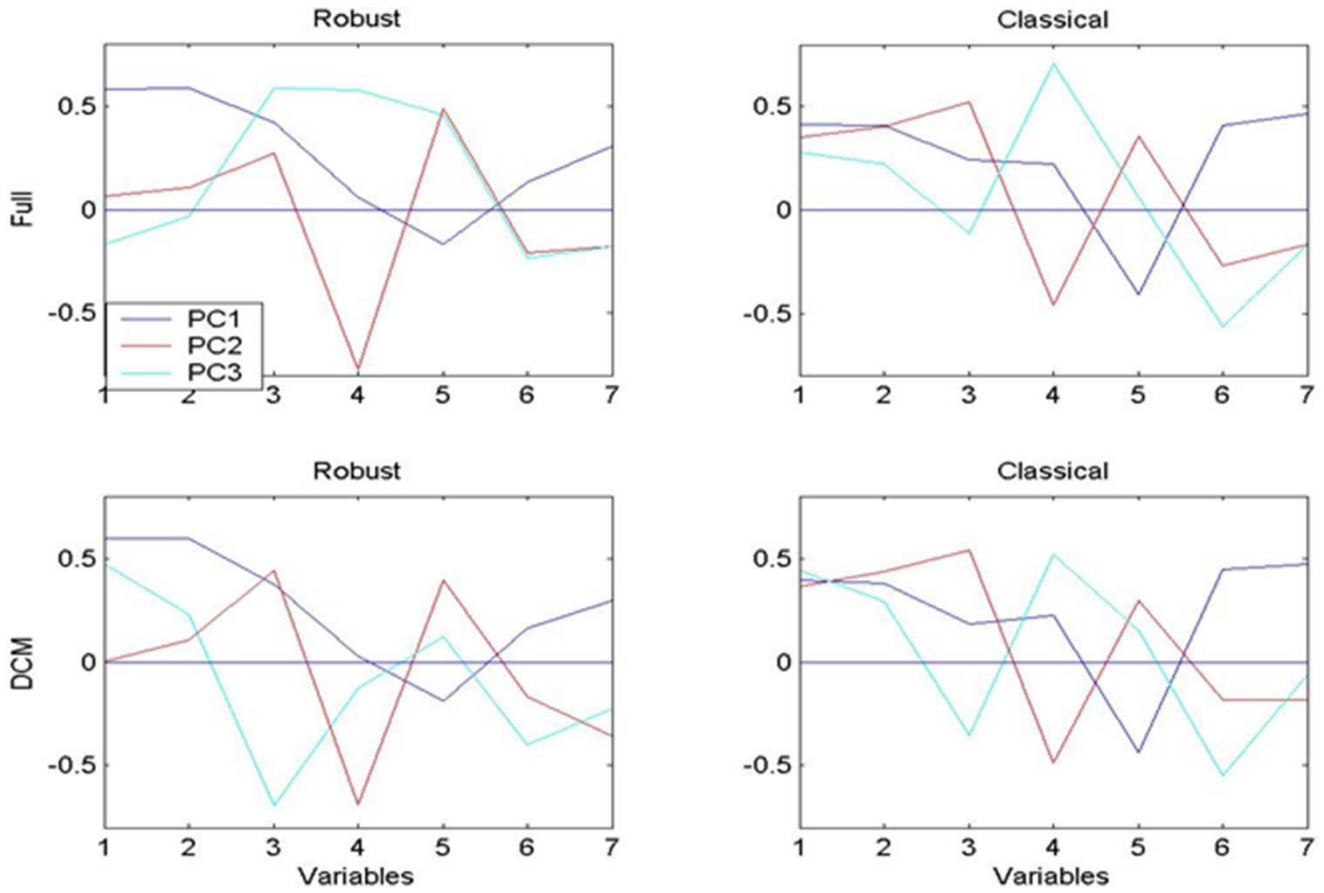
Figure 6. Map representation of the Components 1 (up) and 2 (down) for the PARAFAC model for Antarctica (left) and the Artctic Sea (right).

No of PC	”full data”				”5,10,25,50,100m data”			
	ROBPCA		Classical PCA		ROBPCA		Classical PCA	
	This PC	Total	This PC	Total	This PC	Total	This PC	Total
1	61,26	61,26	62,92	62,92	59.78	59.78	55.67	55.67
2	17,02	78,28	14,26	77,18	19.67	79.46	20.78	76.45
3	8,33	86,61	6,56	83,74	7.04	86.49	8.37	84.81

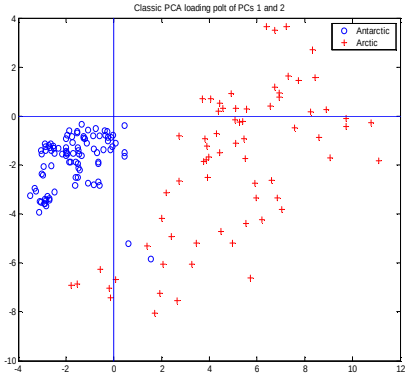
Variance explained (%) by different number of PCs for station-wise unfolded “full” and “selected depth” data;

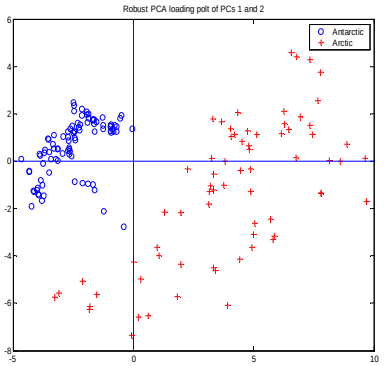
No. of PCs	”Full data”				”5,10,25,50,100m”				”5,10,DCM,50,100m”			
	ROBPCA		Classical PCA		ROBPCA		Classical PCA		ROBPCA		Classical PCA	
	This PC	Total	This PC	Total	This PC	Total	This PC	Total	This PC	Total	This PC	Total
1	61,64	61,64	52,2	52,2	50.98	50.98	50,04	50,04	50.07	50.07	50,10	50,10
2	23,44	85,09	30,96	83,16	34.33	85.31	32,30	82,34	34.43	84.51	33,03	83,13
3	7,11	92,19	8,29	91,46	6.29	91.60	8,79	91,13	5.95	90.46	8,80	91,93

Variance explained (%) by different number of PCs for 3 data sets: “full”, “selected data” and “DCM”;

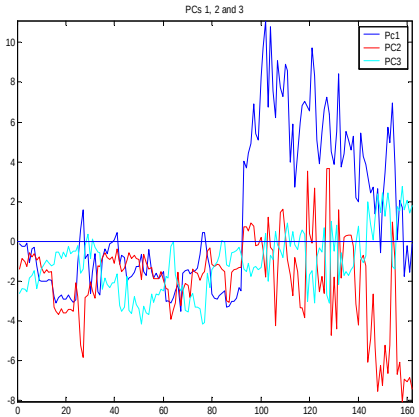


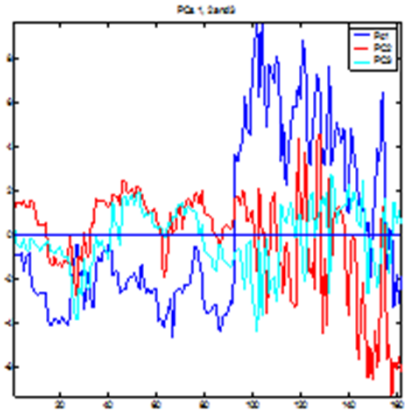
Robust and Classical scores for full and DMC data sets for 3 component PCA model..



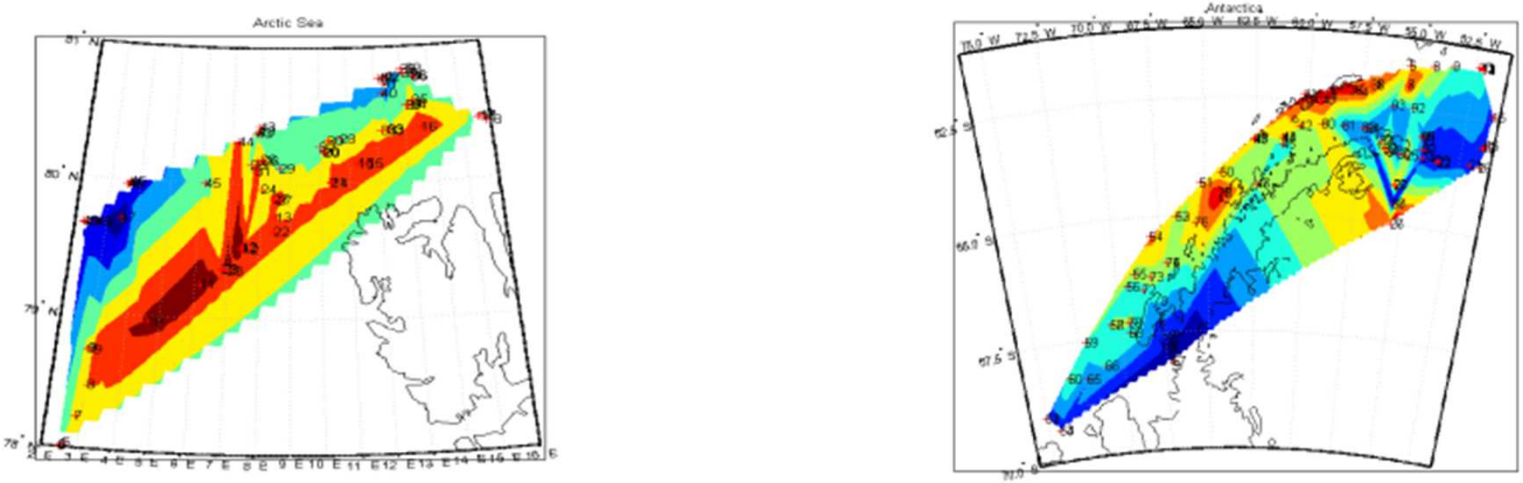


PCA loadings 1 vs 2 for a) classical and b) robust “depth” data;





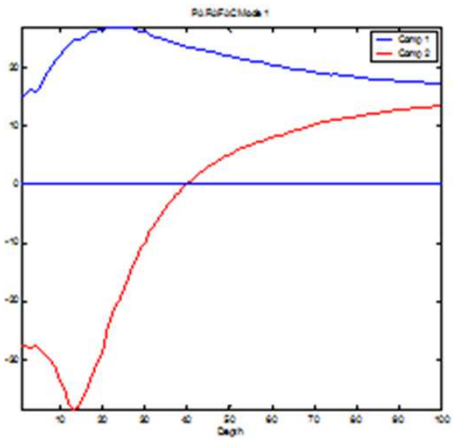
PCA loadings 1, 2 and 3 for a) classical and b) robust “depth” data;

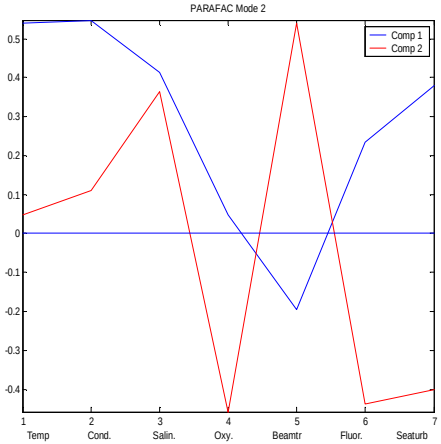


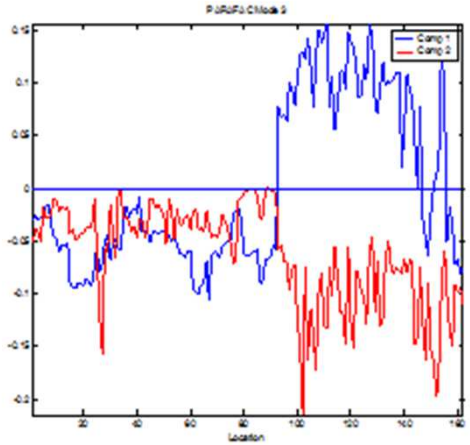
Geo-map of 1st PC for a) Arctic Sea, b) Antarctica

No. of com	”full” dta				”depth”-data			
	Var. expl.	Core cons.	Iter.	Time	Var. expl.	Core cons.	Iter.	Time
1	42,05	100	11	1,52	39.49	100	11	0,3
2	68,01	100	6	0,42	64.22	100	9	0,2
3	76.33	41,17	35	1,42	73.9	-194	9	2,8

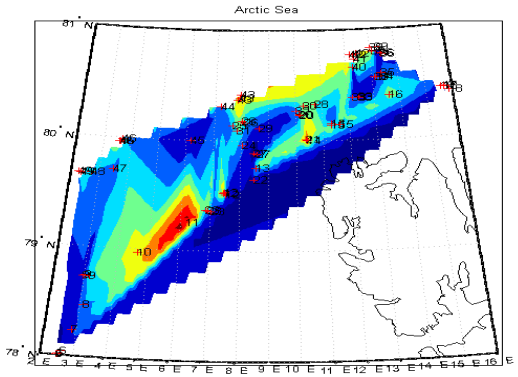
Chosing amount of components in PARAFAC model

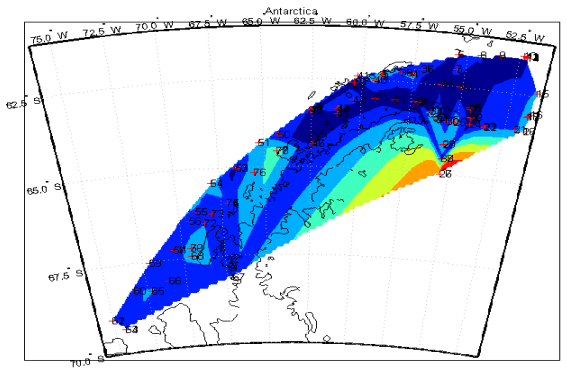






Resolved PARAFAC modes: a) depth, b) variables, c) locations

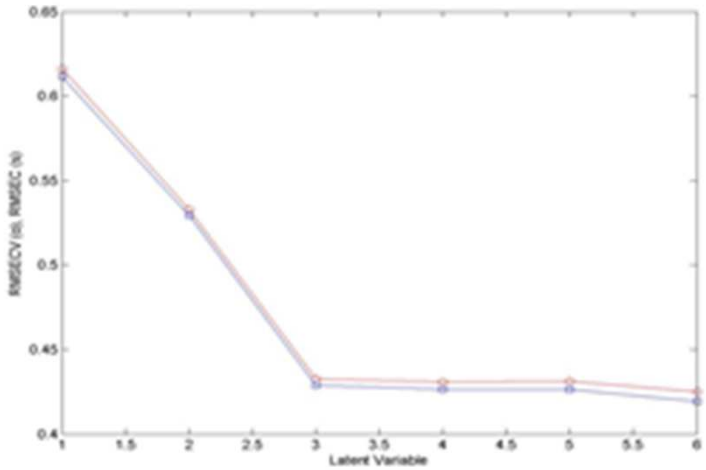




Geo-map of 2nd component for a) Arctic Sea, b) Antartctica

L V.	All variables				Only phisical variables			
	X-Block		Y-Block		X-Block		Y-Block	
	This	Total	This	Total	This	Total	This	Total
1	48,55	48,55	62,60	62,60	52,48	52,48	20,83	20,83
2	32,50	81,05	9,37	71,97	39.91	92,40	4,27	25,10
3	10,47	91,52	9,62	81,60	7,60	100	0,07	25,17

Variance coverage by 3 components PLS



RMSEC and RMSECV for PLS model.