Andrew-made code:

|  |  |
| --- | --- |
| ioneweb: | 2 |
| makelegend | 2 |
| layeredkml | 2 |
| layeredkmllegend | 2 |
| gridtoimage | 2 |
| bytetostr | 2 |
|  |  |
| ionegraphics: | 2 |
| centerfigure | 2 |
| categoryselectmap | 3 |
| selectcategory | 3 |
| mtnicesurfgeneral | 3 |
| mtzoomtocontinent | 5 |
| makeindexed | 5 |
| makeindexedlow | 5 |
| draw5090 | 6 |
| fill5090 | 6 |
| smartsurf | 7 |
| expandaxes | 7 |
|  |  |
| ionedata: | 7 |
| analyzeisland | 7 |
| latlong2rowcol | 8 |
| getlatlon | 8 |
| matrixoffset | 8 |
| mergedata | 8 |
| nearestelement | 9 |
| nearestnonzero | 9 |
| rowcol2latlong | 9 |
| makeunique | 10 |
|  |  |
| misc: | 10 |
| closeto | 10 |
| cmapappend | 10 |
| easyinterp2 | 10 |
| maxval | 10 |
| minval | 10 |
| percentile | 10 |

ioneweb (very specific purpose, mainly useless now): (these codes were used for creating the Atlas Earth web files almost entirely automatically, including most of the HTML code for the web page itself)

webdatascript (automatically writes an HTML file!!!)

% webdatascript - make an html file of ione data

makeLegend (makes a KML file legend)

% MAKELEGEND - make a legend for a KML doc

layeredkml (MAKES A PYRAMID-STRUCTURED GOOGLE EARTH KMZ FILE, part 1)

function layeredkml(image,name)

% LAYEREDKML - create a kmz using a pyramid data structure

%

% Syntax

%

% layeredkml(image,name) where image is an rgb image and name

% serves as a title and a directory name, makes a .kmz of image

layeredkmllegend (MAKES A PYRAMID-STRUCTURED GOOGLE EARTH KMZ FILE, part 2)

% LAYEREDKMLLEGEND - create a kmz using a pyramid data structure

%

% Syntax

%

% layeredkmllegend(data,name) will create a legend for the given

% data 'data' and use 'name' as a directory as well as title

%

% layeredkmllegend(data,name,cmap,folder,shortname,cmin,cmax) lets the

% user additionally specify a colormap, directory, short version of the

% title, and colorbar min and max

gridtoimage (make a world overlay image from a given GIS grid)

function gridtoimage(primarydata,colormap,alphadata,name)

% GRIDTOIMAGE - saves an image at file 'name', using primarydata and

% colormap to create the color and alphadata as alpha.

%This one line uses ind2rgb to cast grid primarydata as an rgb image. It

%has to be rotated and flipped to match image formatting. imwrite is then

%used to save it. imwrite allows an alpha transparency layer, which is

%used to show secondarydata, but there's some calculation and formatting

%involved to make it work.

bytetostr (gives a professional-looking string from the bytes – 1000 bytes becomes “1 KB”, etc)

% BYTETOSTR - returns a string representation of a given number of bytes

ionegraphics:

centerfigure (useful because some of the ionefigures are off-center for some reason – it directly manipulates an image file, automatically finds the part that’s offcenter, and slides it the proper number of pixels to center it)

function O=centerfigure(I)

% centerfigure - fix an off center NiceSurf figure image

%

% SYNTAX

% centerfigure('filename') will load and center image 'filename' and resave

% it while returning the centered image

% centerfigure(image) will return 'image' centered

%

% NOTES

% Only works with NiceSurf and similar figures that aren't too far from the

% default output.

%

% EXAMPLES

% nsg(magic(15));

% outputfig('force','example.png');

% centerfigure('example.png');

%

CategorySelectMap (interactive, engaging data interface. Click on a region and all regions sharing its value will be automatically highlighted over a satellite image of the world. Very very cool.)

% CATEGORYSELECTMAP - interactive category selection tool

%

% Syntax:

% CategorySelectMap(Data,cmap,backdata,bmap);

%

% Creates an interactive figure which lets the user click on a spot on the

% map to highlight it and all other areas with the same value. If provided,

% it will display another data set behind the primary one. (By default, a

% medium-quality composite satellite image of earth is used.) This data set

% can be an indexed image at 2160x4320 pix. To make an image display

% properly on a map projection, use the following:

% CategorySelectMap(Data,cmap,flipud(rot90(backdata)),bmap)

% Higher-resolution colormaps will look much nicer but may also seriously

% impact runtime and space-efficiency.

%

%

% Example

%

% load([iddstring 'YieldGap/AreaFiltered/' ...

% 'YieldGap\_Maize\_MaxYieldPct\_95\_AreaFilteredClimateSpace\_10x10\_prec.mat']);

% data=OS.ClimateMask;

%

% load([iddstring 'YieldGap/AreaFiltered\_Soil/' ...

% 'YieldGap\_Maize\_MaxYieldPct\_95\_AreaFilteredClimateSpaceWithSoil\_5x5\_prec.mat']);

% data=OS.ClimateMask;

% data=data(1:2:end,1:2:end);

% CategorySelectMap(data)

%

SelectCategory

% selectCategory - helper function to CategorySelectMap

mtNiceSurfGeneral (Literally 10x the speed of nsg normal)

% MTNICESURFGENERAL - uberplotting program. Much faster than nsg but no

% borders. Not all the UI tools work.

%

%

% Syntax:

% mtNiceSurfGeneral(Data);

% mtNiceSurfGeneral(Data,STRUCT);

% mtNiceSurfGeneral(Long,Lat,Data,STRUCT);

% mtNiceSurfGeneral(DataStruct,STRUCT);

% mtNiceSurfGeneral(Data,STRUCT,'propertyname','propertyvalue');

% mtNiceSurfGeneral(Data,'propertyname','propertyvalue');

%

% OS=mtNiceSurfGeneral(...) returns OS with calculated fields (e.g.

% coloraxis)

% Struct can have the following fields [Default]

%

% NSS.coloraxis=coloraxis

%

% where coloraxis can have the following forms:

% [] (empty vector)

% coloraxis from data minimum to data maximum

% [f] where f is between 0 and 1

% coloraxis from data minimum to 100\*f percentile maximum

% This syntax useful if there are a few outliers

% [0] coloraxis from -max(abs(Data)) to +max(abs(Data))

% This syntax is useful for aligning 0 with the center of a

% colorbar

% [-f] where f is between 0 and 1

% hybrid of previous two syntaxes. coloraxis centered on 0, and

% will extend to +/- 100\*f percentile absolute maximum

% This syntax useful if there are a few outliers in a plot of

% relative changes.

% [cmin cmax]

% coloraxis from cmin to cmax

%

%

%

% NSS.Units ['']

% NSS.TitleString %or NSS.Title

% NSS.FileName = if 'on' will use titlestring

% NSS.cmap

% NSS.LongLatBox

% NSS.DisplayNotes - this will be placed on the lower left of graph

% NSS.Description - this will be saved as metadata within the file

% NSS.PlotArea='World';

% NSS.coloraxis=[];

% NSS.Description='';

% NSS.DisplayNotes='';

% NSS.uppermap='white'; %or nodatacolor

% NSS.lowermap='emblue'; % or ocean or oceancolor

% NSS.colorbarpercent='off';

% NSS.colorbarfinalplus='off';%

% NSS.colorbarminus='off';%

% NSS.panoplytriangles=[0 0]; % left/right logical turns on L/R triangle

% NSS.eastcolorbar='off';%

% NSS.resolution='-r600';%

% NSS.figfilesave='on';%

% NSS.plotflag='on'; %allows for calling functions to turn off plotting

% NSS.fastplot='off'; %downsamples data/turns off printing for fast plots

% % acceptable values 'on', 'halfdegree'

% NSS.longlatlines='on' %turns lat long grid on or off

% NSS.plotstates='bricnafta' %adm bounds.

% {'off','countries','bricnafta','states','gadm0','gadm1','gadm2'}

% NSS.categorical='off';

% NSS.categoryranges={};

% NSS.categoryvalues={};

% NSS.DataCutoff=9e9;

% NSS.MakePlotDataFile='off';

%

%

% Example:

%

% mtnicesurfgeneral(easyinterp2(magic(15),4320,2160));

%

%

% See Also: NiceSurfGeneral IoneSurf ShowUi HideUi FastSurf

mtZoomToContinent

% MTZOOMTOCONTINENT – zoomtocontinent modified to work with mtnicesurfgeneral

makeindexed (a rip-off of a similar function in the image toolkit – works just as well though)

% MAKEINDEXED - create an indexed version of a given rgb image

% with no loss of quality.

%

% SYNTAX

% [IM CMAP]=MakeIndexed(Image) returns array IM and colormap CMAP

% representing rgb image Image.

%

% Use makeindexedlow if the image quality may be reduced and mem

% is an issue.

%

% EXAMPLE

% image=imread('peppers.png');

% [im cmap]=MakeIndexed(image);

% surface(zeros(size(im)),flipud(im),...

% 'FaceColor','texturemap',...

% 'EdgeColor','none',...

% 'CDataMapping','direct')

% colormap(cmap);

%

% See also

% MakeIndexedLow

makeindexedlow (again, rip-off of an image toolkit function, but works [though it uses a different method of compression])

% MAKEINDEXEDLOW - using simple compression, outputs an indexed image w/

% associated colormap. Slow, but can produce nicely-compressed results.

%

% SYNTAX

% [IM CMAP]=MakeIndexed(Image,q) returns array IM and colormap CMAP

% representing rgb image Image, compressed to degree q (where q is

% the maximum color range to which a value may be assigned - between

% 0 and 1, with 0.001 very low compression and 0.5 very high.

%

% If memory is not an issue or time is, use MakeIndexed.

%

% EXAMPLE

% image=imread('peppers.png');

% [im cmap]=MakeIndexedLow(image,.2);

% surface(zeros(size(im)),flipud(im),...

% 'FaceColor','texturemap',...

% 'EdgeColor','none',...

% 'CDataMapping','direct')

% colormap(cmap);

%

% See Also

% MakeIndexed

draw5090 (very cool display tool; satellite image of Earth with bands showing the 50th and 90th percentiles)

% DRAW5090 draw a figure highlighting data at the 50th and 90th percentiles

%

% fill5090(Data) plots Data on a Robinson projection and fills in all

% cells at the 50th and 90th percentiles of the nonzero points with

% solid color.

%

% Example:

%

% S=OpenNetCDF([iddstring 'Crops2000/crops/maize\_5min.nc'])

% tmp=S.Data(:,:,2); %yield ... not area

% draw5090(tmp);

%

% See also fill5090

fill5090 (this one has the bands filled in)

% FILL5090 draw a figure highlighting data above the 50th and 90th percentiles

%

% fill5090(Data) plots Data on a Robinson projection and fills in all

% cells above the 50th and 90th percentiles of the nonzero points with

% solid color.

%

% Example:

%

% S=OpenNetCDF([iddstring 'Crops2000/crops/maize\_5min.nc'])

% tmp=S.Data(:,:,2); %yield ... not area

% fill5090(tmp);

%

% See also draw5090

smartsurf (quick, easy display tool for sparse data; you don’t have to manually zoom in)

% SMARTSURF - NiceSurfGeneral wrapper with intelligent zoom

%

% SYNTAX

% smartsurf(data) will zoom to the not-NaNs in data

%

% smartsurf(data,S) passes structure S to nsg along with data

%

% EXAMPLE

% smartsurf(easyinterp2(zeros(20)/magic(20)==5,4320,2160))

%

expandaxes (simple fix to nsg’s habit of making tiny map axes)

% EXPANDAXES - expand anq axes to better fill its figure

%

% SYNTAX

% oldposition=expandaxes will set the current axes to position [0 0 1 1]

% and set oldposition to the previous position

% oldposition=expandaxes(A) uses given axes A instead of the current axes

%

% EXAMPLE

% nsg(easyinterp2(magic(5),4320,2160));

% set(gca,'position',[.5 .5 .1 .1]);

% expandaxes

ionedata:

analyzeisland (useful, efficient process to find the contiguous region surrounding a certain index. For instance, used extensively in watershed analysis codes.)

% [rv,cv,area,edge]=analyzeisland(r,c,A) - find the row and col indices,

% area, and perimeter of the contiguous region surrounding a point in an

% array

%

% SYNTAX

% [rv,cv,area,edge]=analyzeisland(r,c,A) returns rv and cv as row and col

% indices vectors, area as the number of points in the contiguous region,

% and edge as the length of the perimeter.

%

% recursion is technically avoided thanks to use of integer vectors to save

% a list of what would otherwise be recursive commands; this means that it

% can run on any size array when otherwise MATLAB would have a recursion

% depth limit error

%

% EXAMPLE

% A=rand(9,9);

% A=floor(A\*3);

% [rv,cv,area,edge]=analyzeisland(5,5,A);

latlong2rowcol (give a row and col of a GIS grid and get back the lat and long.)

% LatLong2RowCol - convert latitude and longitude to row and column

%

% SYNTAX

% LatLong2RowCol(latpos,longpos,Data) will return the row/col

% associated with latpos and longpos in array Data.

%

% LatLong2RowCol(latpos,longpos,Lat,Long) will return the row/col

% associated with latpos/longpos in a map of size Lat/Long or of size

% length(Lat)/length(Long).

%

getlatlon (if given a GIS grid w/o lat/lon vectors, create those vectors. Useful since you don’t have to deal with complex structures – just pull the vectors from the data grid when they’re needed.)

% getlatlon - create lat and lon vectors

%

% SYNTAX

% [lats lons]=getlatlon(data) will return lat and lon vectors matching the

% resolution of data and assuming that data's extent is the entire world

%

% [lats lons]=getlatlon(data,latextent,lonextent) uses latextent and

% lonextent for the extent of data

%

% latextent and lonextent may be ascending or descending. The actual

% content of data is never examined, just its size.

%

% EXAMPLE

% [lats lons]=getlatlon(zeros(5,4))

%

matrixoffset (used in centerfigure as well as mtNiceSurfGeneral [faster to move the data than change the map axes]. Fast, easy to use matrix/array transformation with no MATLAB built in equivalent)

% MATRIXOFFSET - vertically and horizontally moves the data in an array

%

% SYNTAX

% matrixoffset(I,roff,coff) moves every element in I roff rows down and

% coff columns right, wrapping data from one side to the other or top to

% bottom.

%

% matrixoffset(I,roff,coff,fill) instead of wrapping data, fills newly

% vacant spaces with fill.

%

% EXAMPLES

% O=matrixoffset(magic(5),2,-1,0)

%

mergedata (VERY VERY COOL. Given two GIS data sets that don’t necessarily share the same lat/lon extent or resolution, merge one on top of the other.)

% MERGEDATA - pastes one array over another

%

% SYNTAX

% mergedata(background,blon,blat,data,dlon,dlat), where background is some

% background data like a global land mask, with longitude and latitude

% defined by blon and blat, and data is an array to be pasted on top of it

% with latitude and longitude defined by dlon and dlat, will put data over

% background such that the resulting array will have the same extent and

% same lat-long values as background.

% mergedata(background,blon,blat,data,dlon,dlat,method) will use 2D

% interpolation method 'method'

% mergedata(background,data) will do it without lats and longs if the grids

% are the same size

%

% NOTES

% The extent of data must be within the extent of background. NaNs will be

% treated as transparent and will not be pasted.

%

% EXAMPLES

% countries=OpenNetCDF([iddstring ...

% '/AdminBoundary2010/Raster\_NetCDF/1\_Countries\_0.5deg/ctry\_0.5.nc']);

% S=OpenNetCDF([iddstring '/Crops2000/crops/maize\_5min.nc']);

% cropgrid=S.Data(:,:,4)+50000;

% cropgrid(cropgrid>999999|cropgrid<50000)=NaN;

% result=mergedata(countries.Data,countries.Long,countries.Lat, ...

% cropgrid,S.Long,S.Lat,'linear');

%

nearestelement (no apparent builtin MATLAB equivalent – find the array element closest to a certain value)

% NEARESTELEMENT - return the nearest element to x of 1D array A

nearestNonZero (used in several functions including centerfigure.)

% NEARESTNONZERO - return the value of the nearest nonzero element to index r,c

%

% SYNTAX

% v=nearestNonZero(r,c,G) - return the value of the closest nonzero element

% in G to r,c, as determined by expanding a square from r,c and stopping as

% soon as a nonzero element is within the square. If two or more appear in

% the square at once, the one that has the lowest linear index - ie the one

% that is furthest left and, if >1 are in the same column, the one in that

% column that is closer to the top - will be returned.

%

% EXAMPLE

% G=magic(5);

% G=G.\*(G>20);

% v=nearestNonZero(1,1,G);

rowcol2latlong (give a lat and a long and get the row and column corresponding to it in a GIS grid)

% LatLong2RowCol - convert row and column to latitude and longitude

%

% SYNTAX

% RowCol2LatLong(row,col,Data) will return the lat/long

% associated with row and col in array Data.

%

% RowCol2LatLong(row,col,Lat,Long) will return the lat/long

% associated with row/col in a map of size Lat/Long or of size

% length(Lat)/length(Long).

%

% EXAMPLE

% [latpos,longpos]=RowCol2LatLong(5,4,10,20);

makeunique (again, used extensively during watershed analysis – and the latest version is surprisingly quick and efficient. Make every contiguous region its own unique value.)

% MAKEUNIQUE - set each "data island" in data to a unique identifier

%

% SYNTAX

% A=makeunique(data) returns array A of size data with each unique contiguous region set to a unique value

%

% EXAMPLE

% A=makeunique(floor(rand(A)\*3));

misc:

closeto (quick function for finding whether values are close, but not necessarily equal)

% CLOSETO determine approximate equality

%

% Syntax:

%

% a=CLOSETO(x,x1,tol) checks if elements of x are within tol of x1

% (inclusive). Default value of tol is x1/1e-7

%

% Example: find( closeto(10:100,60,6) & closeto(10:100,70,4))

cmapappend (don’t remember exactly how this works, but it involves appending two colormaps in such a way that they color data in a way that makes sense)

% complicated, very specifically designed as helper for AMTSurf

easyinterp2 (VERY USEFUL. If data doesn’t match the right size, this will make it the right size, with much easier syntax than the matlab function interp2)

% EasyInterp2 - interpolate 2-dimensional matrix with easy syntax

%

% SYNTAX

% B=EasyInterp2(A,n); B is A expanded to n times original size

%

% B=EasyInterp2(A,n,method); B is A expanded to n times original size

% using method

%

% B=EasyInterp2(A,rows,cols); B is A stretched to size rows x cols

%

% B=EasyInterp2(A,rows,cols, method); B is A stretched to size rows x

% cols using method

%

%

% EasyInterp2 is designed to make interp2, the built in 2-d matrix

% interpolation function, more accessible. It can be used to stretch a

% small data set to a greater size if needed, or a vice versa, and

% will return the entered matrix exactly as-was if its dimensions match

% rows x cols. Without this function, imresize is the easiest way to do

% this.

%

% EXAMPLE

% B=EasyInterp2(magic(5,5),4320,2160,'linear');

%

maxval

% MAXVAL - return maximum element of A of any dimension

%

% SYNTAX

% m=maxval(A) returns the maximum element of A

%

% EXAMPLE

% m=maxval(rand(5,5,5,5,5,5))

%

minval

% MINVAL - return minimum element of A of any dimension

%

% SYNTAX

% m=minval(A) returns the minimum element of A

%

% EXAMPLE

% m=minval(rand(5,5,5,5,5,5))

%

percentile (use to find not just the min/max values, but values in the middle – quartiles perhaps.)

% percentile - give a value at a certain percentile

%

% SYNTAX

% val=percentile(A,p) returns the value of the p percentile of A or the p\*100

% percentile of A if p is <= 1

%

% EXAMPLE

% val=percentile(1:100,.33333);

%