LSDPlottingTools Documentation

Release

Simon M Mudd

Apr 08, 2017

Contents:

This is the basic documentation that simply repeats information sored within the python files themselves.

CHAPTER 1

LSDPlottingTools package

Submodules

LSDPlottingTools.LSDMap_BasicManipulation module

LSDPlottingTools.LSDMap_BasicManipulation.**BasicMassBalance**(*path*, *file1*, *file2*)

This function checks the difference in "volume" between two rasters.

Parameters

- **path** (*str*) – The path to the files
- **file1** (*str*) – The name of the first raster.
- **file2** (*str*) – The name of the second raster

Returns The differnece in the volume betweeen the two rasters

Return type float

Author: SMM

LSDPlottingTools.LSDMap_BasicManipulation.**BasinKeyToJunction**(*grouped_data_list*, *thisPointData*)

This takes a basin_info_csv file (produced by several LSDTopoTools routies) and spits out lists of the junction numbers (it converts basin numbers to junction numbers).

Parameters

- **grouped_data_list** (*int list*) – A list of list of basin numbers
- **thisPointData** (*str*) – A point data object with the basins

Returns the junction numbers of the basins.

Return type Junction_grouped_list

Author: SMM

LSDPlottingTools.LSDMap_BasicManipulation.**BasinOrderToBasinRenameList**(*basin_order_list*)

When we take data from the basins they will be numbered accoring to their junction rank, which is controlled by flow routing.

The result is often numbered basins that have something that appears random to human eyes. We have developed a routine to renumber these basins. However, the way this works is to find a basin number and rename in the profile plots, such that when it finds a basin number it will rename that. So if you want to rename the seventh basin 0, you need to give a list where the seventh element is 0.

This is a pain because how one would normally order basins would be to look at the image of the basin numbers, and then write the order in which you want those basins to appear.

This function converts between these two lists. You give the function the order you want the basins to appear, and it gives a renaming list.

Parameters **basin_order_list** (*int*) – the list of basins in which you want them to appear in the numbering scheme

Returns The index into the returned basins

Author: SMM

LSDPlottingTools.LSDMap_BasicManipulation.**BasinOrderer**(*thisPointData*, *FileName*, *criteria_string*, *reverse=False*, *exclude_criteria_string=u'None'*, *exlude_criteria_greater=False*, *exclude_criteria_value=0*)

LSDPlottingTools.LSDMap_BasicManipulation.**ConvertNorthingForImshow**(*RasterName*, *Northing*)

This returns a northing that is inverted using the minimum and maximum values from the raster for use in imshow (because imshow inverts the raster)

Parameters

- **RasterName** (*str*) – The raster's name with full path and extension
- **Northing** (*float*) – The northing coordinate in metres (from UTM WGS84)

Returns ConvertedNorthing The northing inverted from top to bottom

Author: SMM

LSDPlottingTools.LSDMap_BasicManipulation.**GetHillshade**(*raster_filename*, *new_raster_filename*, *azimuth=315*, *angle_altitude=45*, *driver_name=u'ENVI'*, *NoDataValue=-9999*)

This calls the hillshade function from the basic manipulation package, but then prints the resulting raster to file.

Parameters

- **raster_filename** (*str*) – The raster's name with full path and extension
- **new_raster_filename** (*str*) – The name of the raster to be printed
- **azimuth** (*float*) – Azimuth angle (compass direction) of the sun (in degrees).
- **angle_altitude** (*float*) – Altitude angle of the sun.
- **driver_name** (*str*) – The raster format (see gdal documentation for options. LSDTopoTools used "ENVI" format.)

- **NoDataValue** – The nodata value. Usually set to -9999.

LSDPlottingTools.LSDMap_BasicManipulation.**GetUTMEastingNorthing**(*EPSG_string*, *latitude*, *longitude*)

This returns the easting and northing for a given latitude and longitide

Parameters

- **ESPG_string** (*str*) – The ESPG code. 326XX is for UTM north and 327XX is for UTM south
- **latitude** (*float*) – The latitude in WGS84
- **longitude** (*float*) – The longitude in WGS84

Returns easting,northing The easting and northing in the UTM zone of your selection

Author: Simon M Mudd

LSDPlottingTools.LSDMap_BasicManipulation.**MaskByCategory**(*rasterArray*, *rasterForMasking*, *data_list*)

This function takes values from an integer raster and renames them based on a list.

It is useful for renaming basin numbers.

Parameters

- **rasterArray** (*np.array*) – The raster array
- **grouped_data_list** (*int*) – A list of lists containing groups to be redefined
- **spread** (*int*) – How big of a difference between groups. For plotting this helps to generate differenc colours.

Returns The new array

Return type np.array

Author: SMM

LSDPlottingTools.LSDMap_BasicManipulation.**NanBelowThreshold**(*rasterArray*, *threshold*)

This function takes an array and turns any element below threshold to a nan

It is useful for renaming basin numbers.

Parameters

- **rasterArray** (*np.array*) – The raster array
- **threshold** (*int*) – The threshold value

Returns The new array

Return type np.array

Author: SMM

LSDPlottingTools.LSDMap_BasicManipulation.**RasterMeanValue**(*path*, *file1*)

This takes the average of a raster.

Parameters

- **path** (*str*) – The path to the raster
- **file1** (*str*) – The name of the file

Returns The mean

Return type mean_value

Author: SMM

LSDPlottingTools.LSDMap_BasicManipulation.**RedefineIntRaster**(*rasterArray*, *grouped_data_list*, *spread*)

This function takes values from an integer raster and renames them based on a list.

It is useful for renaming basin numbers.

Parameters

- **rasterArray** (*np.array*) – The raster array
- **grouped_data_list** (*int*) – A list of lists containing groups to be redefined
- **spread** (*int*) – How big of a difference between groups. For plotting this helps to generate differenc colours.

Returns The new array

Return type np.array

Author: SMM

LSDPlottingTools.LSDMap_BasicManipulation.**SetNoDataBelowThreshold**(*raster_filename*, *new_raster_filename*, *threshold=0*, *driver_name=u'ENVI'*, *NoDataValue=-9999*)

This takes a raster an then converts all data below a threshold to nodata, it then prints the resulting raster.

Parameters

- **raster_filename** (*str*) – The raster's name with full path and extension
- **new_raster_filename** (*str*) – The name of the raster to be printed
- **threshold** (*float*) – Data below this in the original raster will be converted to nodata.
- **driver_name** (*str*) – The raster format (see gdal documentation for options. LSDTopoTools used "ENVI" format.)
- **NoDataValue** (*float*) – The nodata value. Usually set to -9999.

Returns None, but prints a new raster to file

Author: SMM

LSDPlottingTools.LSDMap_BasicManipulation.**SetToConstantValue**(*raster_filename*, *new_raster_filename*, *constant_value*, *driver_name=u'ENVI'*)

This takes a raster an then converts all non-nodata to a constant value.

This is useful if you want to make masks, for example to have blocks of single erosion rates for cosmogenic calculations.

Parameters

- **raster_filename** (*str*) – The raster's name with full path and extension

- **new_raster_filename** (*str*) – The name of the raster to be printed
- **constant_value** (*float*) – All non-nodata will be converted to this value in a new raster.
- **driver_name** (*str*) – The raster format (see gdal documentation for options. LSDTopoTools used "ENVI" format.)
- **NoDataValue** (*float*) – The nodata value. Usually set to -9999.

Returns None, but prints a new raster to file

Author: SMM

LSDPlottingTools.LSDMap_BasicManipulation.**SimpleSwath**(*path*, *file1*, *axis*)

This function averages all the data along one of the directions

Parameters

- **path** (*str*) – The path to the files
- **file1** (*str*) – The name of the first raster.
- **axis** (*int*) – Either 0 (rows) or 1 (cols)

Returns

A load of information about the swath.

- means
- medians
- std_deviations
- twentyfifth_percentile
- seventyfifth_percentile

at each node across the axis of the swath.

Return type float

Author: SMM

LSDPlottingTools.LSDMap_BasicPlotting module

LSDPlottingTools.LSDMap_BasicPlotting.**BasicDensityPlot**(*FileName*, *thiscmap=u'gray'*, *colorbarlabel=u'Elevation in meters'*, *clim_val=(0, 0)*, *FigFileName=u'Image.pdf'*, *FigFormat=u'show'*, *size_format=u'esurf'*, *is_log=False*)

This creates a plot of a raster. The most basic plotting function. It uses AxisGrid to ensure proper placment of the raster.

Parameters

- **FileName** (*str*) – The name of the raster (with full path and extension).
- **thiscmap** (*colormap*) – The colourmap to be used.

- **colorbarlabel** (*str*) – The label of the colourbar
- **clim_val** (*float, float*) – The colour limits. If (0,0) then the min and max raster values are used.
- **FigFilename** (*str*) – The name of the figure (with extension)
- **FigFormat** (*str*) – the format of the figure (e.g., jpg, png, pdf). If "show" then the figure is plotted to screen.
- **size_format** (*str*) – the size of the figure
- **islog** (*bool*) – True if you want the figure to have a log density not the density

Returns A density plot of the raster

Author: Simon M Mudd

LSDPlottingTools.LSDMap_BasicPlotting.**BasicDrapedPlotGridPlot**(*FileName*, *DrapeName*, *thiscmap=u'gray'*, *drape_cmap=u'gray'*, *colorbarlabel=u'Elevation in meters'*, *clim_val=(0, 0)*, *drape_alpha=0.6*, *FigFileName=u'Image.pdf'*, *FigFormat=u'show'*, *dpi_save=250*)

This creates a draped plot of a raster. It uses AxisGrid to ensure proper placment of the raster.

Parameters

- **FileName** (*str*) – The name of the raster (with full path and extension).
- **DrapeName** (*str*) – The name of the drape raster (with full path and extension). If the DrapeName is "None" it will calculate the hillshade.
- **thiscmap** (*colormap*) – The colourmap to be used.
- **drape_cmap** (*colormap*) – The colourmap to be used for the drape.
- **colorbarlabel** (*str*) – The label of the colourbar
- **clim_val** (*float, float*) – The colour limits. If (0,0) then the min and max raster values are used.
- **drape_alpha** (*float*) – The alpha value (transparency) of the drape
- **FigFilename** (*str*) – The name of the figure (with extension)
- **FigFormat** (*str*) – the format of the figure (e.g., jpg, png, pdf). If "show" then the figure is plotted to screen.

Returns A density plot of the draped raster

Author: Simon M Mudd

LSDPlottingTools.LSDMap_BasicPlotting.**BasinsOverFancyHillshade**(*FileName*, *HSName*, *BasinName*, *Basin_csv_name*, *basin_point_data*, *thiscmap=u'gray'*, *drape_cmap=u'gray'*, *clim_val=(0, 0)*, *drape_alpha=0.6*, *FigFileName=u'Image.pdf'*, *FigFormat=u'show'*, *elevation_threshold=0*, *grouped_basin_list=[]*, *basin_rename_list=[]*, *spread=20*, *chanPointData=u'None'*, *label_sources=False*, *source_chi_threshold=10*, *size_format=u'esurf'*)

This creates a plot with a hillshade draped over elevation (or any other raster) with the basins on them.

Parameters

- **FileName** (*str*) – The name of the raster (with full path and extension).
- **HSName** (*str*) – The name of the hillshade raster (with full path and extension).
- **BasinName** (*str*) – The name of the basin raster (with full path and extension).
- **Basin_csv_name** (*str*) – The name of the csv file where basin info is stored
- **basin_point_data** (LSDMap_PointData) – The basin point data
- **thiscmap** (*colormap*) – The colourmap to be used.
- **drape_cmap** (*colormap*) – The colourmap to be used for the drape.
- **clim_val** (*float, float*) – The colour limits. If (0,0) then the min and max raster values are used.
- **drape_alpha** (*float*) – The alpha value (transparency) of the drape
- **FigFilename** (*str*) – The name of the figure (with extension)
- **FigFormat** (*str*) – the format of the figure (e.g., jpg, png, pdf). If "show" then the figure is plotted to screen.
- **elevation_threshold** (*float*) – If raster values are less than this threshold they become nodata.
- **grouped_basin_list** (*int list*) – A list of lists with basins to be grouped.
- **basin_rename_list** (*int list*) – A list of updated names for the basins. So if you wanted basin 4 to be renamed basin 6 the fourth element in this list would be 6.

- **spread** (*float*) – Basins get a different number each, this is the spread between groups that controls how different the grouped basins are.
- **chanPointData** (*str ir LSDMap_PointData*) – Either "none" or a point data object with the channel network
- **label_sources** (*bool*) – Wheteher or not to label the sources
- **source_chi_threshold** (*float*) – Sources with length less than this will be plotted.
- **size_format** (*str*) – Can be "big" (16 inches wide), "geomorphology" (6.25 inches wide), or "ESURF" (4.92 inches wide) (defualt esurf).

Returns A density plot of the draped raster

Author: SMM

LSDPlottingTools.LSDMap_BasicPlotting.**DrapedOverFancyHillshade**(*FileName*, *HSName*, *DrapeName*, *thiscmap=u'gray'*, *drape_cmap=u'gray'*, *colorbarlabel=u'Basin number'*, *clim_val=(0, 0)*, *drape_alpha=0.6*, *FigFileName=u'Image.pdf'*, *FigFormat=u'show'*, *elevation_threshold=0*)

This creates a draped plot of a raster. It uses AxisGrid to ensure proper placment of the raster. It also includes a hillshde to make the figure look nicer (so there are three raster layers). In this case you need to tell it the name of the hillshade raster.

Parameters

- **FileName** (*str*) – The name of the raster (with full path and extension).
- **HSName** (*str*) – The name of the hillshade raster (with full path and extension).
- **DrapeName** (*str*) – The name of the drape raster (with full path and extension).
- **thiscmap** (*colormap*) – The colourmap to be used.
- **drape_cmap** (*colormap*) – The colourmap to be used for the drape.
- **colorbarlabel** (*str*) – The label of the colourbar
- **clim_val** (*float, float*) – The colour limits. If (0,0) then the min and max raster values are used.
- **drape_alpha** (*float*) – The alpha value (transparency) of the drape
- **FigFilename** (*str*) – The name of the figure (with extension)
- **FigFormat** (*str*) – the format of the figure (e.g., jpg, png, pdf). If "show" then the figure is plotted to screen.

- **elevation_threshold** (*float*) – If raster values are less than this threshold they become nodata.

Returns A density plot of the draped raster

Author: SMM

LSDPlottingTools.LSDMap_BasicPlotting.**DrapedOverHillshade**(*FileName*, *DrapeName*, *thiscmap=u'gray'*, *drape_cmap=u'gray'*, *colorbarlabel=u'Elevation in meters'*, *clim_val=(0, 0)*, *drape_alpha=0.6*, *ShowColorbar=False*, *ShowDrapeColorbar=False*, *drape_cbarlabel=None*)

This creates a draped plot of a raster.

It uses AxisGrid to ensure proper placment of the raster. It also includes a hillshde to make the figure look nicer (so there are three raster layers).

Note: Remember, this has THREE layers: a base layer, a hillshade and a drape.

Parameters

- **FileName** (*str*) – The name of the raster (with full path and extension).
- **DrapeName** (*str*) – The name of the drape raster (with full path and extension).
- **thiscmap** (*colormap*) – The colourmap to be used.
- **drape_cmap** (*colormap*) – The colourmap to be used for the drape.
- **colorbarlabel** (*str*) – The label of the colourbar
- **clim_val** (*float, float*) – The colour limits. If (0,0) then the min and max raster values are used.
- **drape_alpha** (*float*) – The alpha value (transparency) of the drape
- **ShowColorbar** (*bool*) – Whether you want to show the colorbar
- **drape_cbarlabel** (*str*) – The label of the drape colourbar

Returns A density plot of the draped raster

Author: SMM and DAV

LSDPlottingTools.LSDMap_BasicPlotting.**DrapedOverHillshade_Categories**(*FileName*, *DrapeName*, *nCategories*, *category_min_max*, *thiscmap=u'gray'*, *drape_cmap=u'gray'*, *clim_val=(0, 0)*, *drape_alpha=0.6*, *ShowDrapeColorbar=False*, *drape_cbarlabel=None*, *category_labels=None*)

This creates a draped plot of a categorical raster.

It uses AxisGrid to ensure proper placment of the raster. It also includes a hillshade to make the figure look nicer (so there are three raster layers).

Note: Remember, this has THREE layers: a base layer, a hillshade and a drape.

Parameters

- **FileName** (*str*) – The name of the raster (with full path and extension).
- **DrapeName** (*str*) – The name of the drape raster (with full path and extension).
- **nCategories** (*int*) – The number of categories in the dataset
- **category_min_max** (*float, float*) – The min and max values in the categorical raster.
- **thiscmap** (*colormap*) – The colourmap to be used for the hillshade.
- **drape_cmap** (*colormap*) – The colourmap to be used for the drape.
- **clim_val** (*float,float*) – The colour limits. If (0,0) then the min and max raster values are used.
- **drape_alpha** (*float*) – The alpha value (transparency) of the drape.
- **ShowDrapeColorbar** (*bool*) – Toggles the display of a categorical colorbar.
- **drape_cbarlabel** (*str*) – The label of the drape colourbar.
- **category_labels** (*list*) – List of strings used as category labels.

Returns A density plot of the draped categorical raster

Author: SWDG, after SMM and DAV

LSDPlottingTools.LSDMap_BasicPlotting.**GetTicksForUTM**(*FileName*, *x_max*, *x_min*, *y_max*, *y_min*, *n_target_tics*)

This fuction is used to set tick locations for UTM maps. It tries to optimise the spacing of these ticks.

Parameters

- **x_min** (*float*) – The minimum value on the x axis (in metres).
- **x_max** (*float*) – The maximum value on the x axis (in metres).
- **y_min** (*float*) – The minimum value on the y axis (in metres).
- **y_max** (*float*) – The maximum value on the y axis (in metres).
- **n_target_ticks** (*int*) – The number of ticks you want on the axis (this is optimised so you may not get exactly this number)

Returns List of locations of the ticks in metres. new_x_labels (str list): List of strings for ticks, will be location in kilometres. new_ylocs (float list): List of locations of the ticks in metres. new_y_labels (str list): List of strings for ticks, will be location in kilometres.

Return type new_xlocs (float list)

Author: SMM

LSDPlottingTools.LSDMap_BasicPlotting.**GetTicksForUTMNoInversion**(*FileName*, *x_max*, *x_min*, *y_max*, *y_min*, *n_target_tics*)

This fuction is used to set tick locations for UTM maps. It tries to optimise the spacing of these ticks.

Parameters

- **x_min** (*float*) – The minimum value on the x axis (in metres).
- **x_max** (*float*) – The maximum value on the x axis (in metres).
- **y_min** (*float*) – The minimum value on the y axis (in metres).
- **y_max** (*float*) – The maximum value on the y axis (in metres).
- **n_target_ticks** (*int*) – The number of ticks you want on the axis (this is optimised so you may not get exactly this number)

Returns List of locations of the ticks in metres. new_x_labels (str list): List of strings for ticks, will be location in kilometres. new_ylocs (float list): List of locations of the ticks in metres. new_y_labels (str list): List of strings for ticks, will be location in kilometres.

Return type new_xlocs (float list)

Author: SMM

LSDPlottingTools.LSDMap_BasicPlotting.**Hillshade**(*raster_file*, *azimuth=315*, *angle_altitude=45*, *NoDataValue=-9999*, *z_factor=1*)

Creates a hillshade raster

Parameters

- **raster_file** (*str*) – The name of the raster file with path and extension.
- **azimuth** (*float*) – Azimuth of sunlight
- **angle_altitude** (*float*) – Angle altitude of sun
- **NoDataValue** (*float*) – The nodata value of the raster

Returns The hillshade array

Return type HSArray (numpy.array)

Author: DAV and SWDG

LSDPlottingTools.LSDMap_BasicPlotting.**LongitudinalSwathAnalysisPlot**(*full_file_path*, *ax*)

Longitudinal channel swath profiles from the swath analysis driver output.

Author: DAV & DTM

LSDPlottingTools.LSDMap_BasicPlotting.**MultiLongitudinalSwathAnalysisPlot**(*data_dir*, *wildcard_fname*, *maximum=0*)

For multiple overlaid channel swath profiles.

Parameters

- **data_dir** – Full path to the directory containing the swath profile text files.
- **wildcard_fname** – Any string that can be expanded by python's *glob* to give a list of files, e.g. "some_common_prefix_*.txt"
- **maximum** (*optional*) – Hacky solution, but give this a float value and it will plot the 'zero' line on your swath profile. C.f. maximum length of channel)

Author: DAV

LSDPlottingTools.LSDMap_BasicPlotting.**SwathPlot**(*path*, *filename*, *axis*)

A function that creates a swath in either the x or y direction only. Averages across entire DEM. Exceedingly basic.

Parameters

- **path** (*str*) – the path to the raster
- **filename** (*str*) – the name of the file
- **axis** (*int*) – if 0, swath along x-axis, if not swath along y-axis

Returns A plot of the swath

Author: SMM

LSDPlottingTools.LSDMap_BasicPlotting.**TickConverter**(*x_min*, *x_max*, *n_target_tics*)

This function is used to convert ticks in metres to ticks in kilometres.

Parameters

- **x_min** (*float*) – The minimum value on the axis (in metres).
- **x_max** (*float*) – The maximum value on the axis (in metres)
- **n_target_ticks** (*int*) – The number of ticks you want on the axis (this is optimised so you may not get exactly this number)

Returns List of locations of the ticks in metres. new_x_labels (str list): List of strings for ticks, will be location in kilometres.

Return type new_xlocs (float list)

Author: Simon M Mudd

LSDPlottingTools.LSDMap_BasicPlotting.**TickLabelShortenizer**(*labels*, *n_hacked_digits*)

This takes a list of labels and hacks off digits so you can go, say, from metres to km.

Parameters

- **labels** (*str*) – A list of labels
- **n_hacked_digits** (*int*) – The number of digits to hack off

Returns The new labels

Author: SMM

LSDPlottingTools.LSDMap_BasicPlotting.**TickSpineFormatter**(*ax*, *sizeformat=u'esurf'*)

This formats the line weights on the bounding box and ticks.

Parameters

- **ax1** (*axis object*) – the matplotlib axis object
- **size_format** (*str*) – The size format. Can be geomorhpology, esurf or big

Returns The axis object

Author: SMM

LSDPlottingTools.LSDMap_BasicPlotting.**function_sketcher**(*formula*, *x_range*)

Creates x and y values over specified range for a given lambda function.

Parameters

- **formula** – a lambda function describing a mathematical function example: "lambda x: x**3+2*x-4" (ommit the quotations when passing)

  would represent the line $y = x^3 + 2*x - 4$.

- **x_range** – A range of numbers to calculate y from = range(0,10) or np.linspace

Returns the x coordinates of the function graph as a numpy array (same as x_range) y: the y coordinates of the function graph as a numpy array

Return type x

Author: DAV

LSDPlottingTools.LSDMap_BasicPlotting.**init_plotting_DV**()

Initial plotting parameters.

Author: DAV

LSDPlottingTools.LSDMap_BasicPlotting.**round_to_n**(*x*, *n*)

A rounding function

Parameters

- **x** (*float*) – The number to be rounded
- **n** (*int*) – The number of digits to be rounded to.

Returns The rounded number

Return type Rounded (float)

Author: SMM

LSDPlottingTools.LSDMap_ChiPlotting module

LSDPlottingTools.LSDMap_ChiPlotting.**BasicChannelPlotGridPlotCategories**(*FileName*, *DrapeName*, *chi_csv_fname*, *thiscmap=u'gray'*, *drape_cmap=u'gray'*, *colorbarlabel=u'Elevation in meters'*, *clim_val=(0, 0)*, *drape_alpha=0.6*, *FigFileName=u'Image.pdf'*, *FigFormat=u'show'*, *elevation_threshold=0*, *data_name=u'source_key'*, *source_thinning_threshold=0*, *size_format=u'ESURF'*)

This plots the channels over a draped plot, colour coded by source

Parameters

- **FileName** (*str*) – The name (with full path and extension) of the DEM.
- **DrapenName** (*str*) – The name (with full path and extension) of the drape file (usually a hillshade, but could be anything)
- **chi_csv_fname** (*str*) – The name (with full path and extension) of the cdv file with chi, chi slope, etc information. This file is produced by the chi_mapping_tool.
- **thiscmap** (*colormap*) – The colourmap for the elevation raster
- **drape_cmap** (*colormap*) – The colourmap for the drape raster
- **colorbarlabel** (*str*) – the text label on the colourbar.

- **clim_val** (*float,float*) – The colour limits for the drape file. If (0,0) it uses the minimum and maximum values of the drape file. Users can assign numbers to get consistent colourmaps between plots.
- **drape_alpha** (*float*) – The alpha value of the drape
- **FigFileName** (*str*) – The name of the figure file
- **FigFormat** (*str*) – The format of the figure. Usually 'png' or 'pdf'. If "show" then it calls the matplotlib show() command.
- **elevation_threshold** (*float*) – elevation_threshold chi points below this elevation are removed from plotting.
- **data_name** (*str*) –
- **source_thinning_threshold** (*float*) –
- **size_format** (*str*) – Can be "big" (16 inches wide), "geomorphology" (6.25 inches wide), or "ESURF" (4.92 inches wide) (defualt esurf).

Returns Prints a plot to file.

Author: Simon M. Mudd

LSDPlottingTools.LSDMap_ChiPlotting.**BasicChiCoordinatePlot**(*FileName*, *DrapeName*, *csvfile*, *thiscmap=u'gray'*, *drape_cmap=u'gray'*, *colorbarlabel=u'$\\chi (m)$'*, *clim_val=(0, 0)*, *basin_order_list=[]*, *basin_point_data=u'None'*, *basin_raster_name=u'None'*, *drape_alpha=0.6*, *FigFileName=u'Image.pdf'*, *FigFormat=u'show'*, *size_format=u'ESURF'*)

This plots the chi coordinate, mimicking Sean Willet et al's plots

Parameters

- **FileName** (*str*) – The name (with full path and extension) of the DEM.
- **DrapenName** (*str*) – The name (with full path and extension) of the drape file (usually a hillshade, but could be anything)
- **thisPointData** (LSDMap_PointData) – The point data object with the basic chi points
- **thiscmap** (*colormap*) – The colourmap for the elevation raster
- **drape_cmap** (*colormap*) – The colourmap for the drape raster
- **colorbarlabel** (*str*) – the text label on the colourbar.
- **clim_val** (*float,float*) – The colour limits for the drape file. If (0,0) it uses the minimum and maximum values of the drape file. Users can assign numbers to get consistent colourmaps between plots.
- **basin_order_list** (*list of int*) – The basin indices to be selected

- **basin_point_data** (*LSDM_PointData*) – The mapping between junctions and indices
- **basin_raster_name** (*str*) – If a basin raster name is supplied the chi raster will be masked
- **drape_alpha** (*float*) – The alpha value of the drape
- **FigFileName** (*str*) – The name of the figure file
- **FigFormat** (*str*) – The format of the figure. Usually 'png' or 'pdf'. If "show" then it calls the matplotlib show() command.
- **size_format** (*str*) – Can be "big" (16 inches wide), "geomorphology" (6.25 inches wide), or "ESURF" (4.92 inches wide) (defualt esurf).

Returns Prints a plot to file.

Author: Simon M. Mudd

LSDPlottingTools.LSDMap_ChiPlotting.**BasicChiPlotGridPlot**(*FileName*, *DrapeName*, *chi_csv_fname*, *thisPointData*, *thiscmap=u'gray'*, *drape_cmap=u'gray'*, *colorbarlabel=u'log$_{10}k_{sn}$'*, *clim_val=(0, 0)*, *drape_alpha=0.6*, *FigFileName=u'Image.pdf'*, *FigFormat=u'show'*, *elevation_threshold=0*, *size_format=u'ESURF'*, *dpi_save=750*)

This is the main chi plotting script that prints a chi steepness map over the hillshade. Note that the colour scale for the chi slope values are always cubehelix

Parameters

- **FileName** (*str*) – The name (with full path and extension) of the DEM.
- **DrapenName** (*str*) – The name (with full path and extension) of the drape file (usually a hillshade, but could be anything)
- **chi_csv_fname** (*str*) – The name (with full path and extension) of the cdv file with chi, chi slope, etc information. This file is produced by the chi_mapping_tool.
- **thiscmap** (*colormap*) – The colourmap for the elevation raster
- **drape_cmap** (*colormap*) – The colourmap for the drape raster
- **colorbarlabel** (*str*) – the text label on the colourbar.
- **clim_val** (*float,float*) – The colour limits for the drape file. If (0,0) it uses the minimum and maximum values of the drape file. Users can assign numbers to get consistent colourmaps between plots.
- **drape_alpha** (*float*) – The alpha value of the drape
- **FigFileName** (*str*) – The name of the figure file
- **FigFormat** (*str*) – The format of the figure. Usually 'png' or 'pdf'. If "show" then it calls the matplotlib show() command.

- **elevation_threshold** (*float*) – elevation_threshold chi points below this elevation are removed from plotting.
- **size_format** (*str*) – Can be "big" (16 inches wide), "geomorphology" (6.25 inches wide), or "ESURF" (4.92 inches wide) (defualt esurf).

Returns Prints a plot to file.

Author: Simon M. Mudd

LSDPlottingTools.LSDMap_ChiPlotting.**BasicChiPlotGridPlotKirby**(*FileName*, *DrapeName*, *chi_csv_fname*, *thiscmap=u'gray'*, *drape_cmap=u'gray'*, *colorbarlabel=u'Elevation in meters'*, *clim_val=(0, 0)*, *drape_alpha=0.6*, *FigFileName=u'Image.pdf'*, *FigFormat=u'show'*, *elevation_threshold=0*, *size_format=u'ESURF'*, *dpi_save=500*)

This function plots the chi slope on a shaded relief map. It uses the Kirby and Whipple colour scheme.

Parameters

- **FileName** (*str*) – The name (with full path and extension) of the DEM.
- **DrapenName** (*str*) – The name (with full path and extension) of the drape file (usually a hillshade, but could be anything)
- **chi_csv_fname** (*str*) – The name (with full path and extension) of the cdv file with chi, chi slope, etc information. This file is produced by the chi_mapping_tool.
- **thiscmap** (*colormap*) – The colourmap for the elevation raster
- **drape_cmap** (*colormap*) – The colourmap for the drape raster
- **colorbarlabel** (*str*) – the text label on the colourbar.
- **clim_val** (*float,float*) – The colour limits for the drape file. If (0,0) it uses the minimum and maximum values of the drape file. Users can assign numbers to get consistent colourmaps between plots.
- **drape_alpha** (*float*) – The alpha value of the drape
- **FigFileName** (*str*) – The name of the figure file
- **FigFormat** (*str*) – The format of the figure. Usually 'png' or 'pdf'. If "show" then it calls the matplotlib show() command.
- **elevation_threshold** (*float*) – elevation_threshold chi points below this elevation are removed from plotting.

- **size_format** (*str*) – Can be "big" (16 inches wide), "geomorphology" (6.25 inches wide), or "ESURF" (4.92 inches wide) (defualt esurf).

Returns Does not return anything but makes a plot.

Author: SMM

LSDPlottingTools.LSDMap_ChiPlotting.**ChiProfiles**(*chi_csv_fname*, *FigFileName=u'Image.pdf'*, *FigFormat=u'show'*, *basin_order_list=[]*, *basin_rename_list=[]*, *label_sources=False*, *elevation_threshold=0*, *source_thinning_threshold=0*, *plot_M_chi=False*, *size_format=u'ESURF'*, *plot_segments=False*)

This function plots the chi vs elevation: lumps everything onto the same axis. This tends to make a mess.

Parameters

- **chi_csv_fname** (*str*) – The name (with full path and extension) of the cdv file with chi, chi slope, etc information. This file is produced by the chi_mapping_tool.
- **FigFileName** (*str*) – The name of the figure file
- **FigFormat** (*str*) – The format of the figure. Usually 'png' or 'pdf'. If "show" then it calls the matplotlib show() command.
- **basin_order_list** (*int list*) – The basins to plot
- **basin_rename_list** (*int list*) – A list for naming substitutions
- **label_sources** (*bool*) – If true, label the sources.
- **elevation_threshold** (*float*) – elevation_threshold chi points below this elevation are removed from plotting.
- **source_thinning_threshold** (*float*) –
- **plot_MChi** (*bool*) – If true, plots chi against MChi
- **size_format** (*str*) – Can be "big" (16 inches wide), "geomorphology" (6.25 inches wide), or "ESURF" (4.92 inches wide) (defualt esurf).

Returns Does not return anything but makes a plot.

Author: SMM

LSDPlottingTools.LSDMap_ChiPlotting.**ConvertBasinIndexToJunction**(*BasinPointData*, *BasinIndexList*)

This transforms a basin index list (simply the order of the basins, starting from low to high junction number) to a junction list.

This allows users to go between basin rasters (with junctions listed) and the simpler basin indexing system (which is sequential)

Parameters

- **BasinPointData** (LSDMap_PointData) – a point data object
- **BasinIndexList** (*list of ints*) – The basin indices to be converted to junctions

Returns A list on ints with the basin junctions

Author: SMM

LSDPlottingTools.LSDMap_ChiPlotting.**FindShortSourceChannels**(*these_source_nodes*, *threshold_length*)

This function gets the list of sources that are shorter than a threshold value

Parameters

- **these_source_nodes** (*dict*) – A dict from the FindSourceInformation module
- **threshold_length** (*float*) – The threshold of chi lenght of the source segment

Returns A list of integers of source with the appropriate length

Return type long_sources

Author: SMM

LSDPlottingTools.LSDMap_ChiPlotting.**FindSourceInformation**(*thisPointData*)

This function finds the source locations, with chi elevation, flow distance, etc.

Parameters **thisPointData** (LSDMap_PointData) –

Returns A dict with key of the source node that returns a dict that has the FlowDistance, Chi, and Elevation of each source. Used for plotting source numbers on profile plots.

Author: SMM

LSDPlottingTools.LSDMap_ChiPlotting.**StackedChiProfiles**(*chi_csv_fname*, *FigFileName=u'Image.pdf'*, *FigFormat=u'show'*, *elevation_threshold=0*, *first_basin=0*, *last_basin=0*, *basin_order_list=[]*, *basin_rename_list=[]*, *X_offset=5*, *label_sources=False*, *source_thinning_threshold=0*, *size_format=u'ESURF'*)

This function plots the chi vs elevation: It stacks profiles (so the basins are spaced out) and colours them by the source number.

Parameters

- **chi_csv_fname** (*str*) – The name (with full path and extension) of the cdv file with chi, chi slope, etc information. This file is produced by the chi_mapping_tool.
- **FigFileName** (*str*) – The name of the figure file
- **FigFormat** (*str*) – The format of the figure. Usually 'png' or 'pdf'. If "show" then it calls the matplotlib show() command.
- **elevation_threshold** (*float*) – elevation_threshold chi points below this elevation are removed from plotting.
- **first_basin** (*int*) – The basin to start with (but overridden by the basin list)
- **last_basin** (*int*) – The basin to end with (but overridden by the basin list)
- **basin_order_list** (*int list*) – The basins to plot
- **basin_rename_list** (*int list*) – A list for naming substitutions. Useful because LSDTopoTools might number basins in a way a human wouldn't, so a user can intervene in the names.

- **X_offset** (*float*) – The offest in chi between the basins along the x-axis. Used to space out the profiles so you can see each of them.
- **label_sources** (*bool*) – If true, label the sources.
- **source_thinning_threshold** (*float*) –
- **size_format** (*str*) – Can be "big" (16 inches wide), "geomorphology" (6.25 inches wide), or "ESURF" (4.92 inches wide) (defualt esurf).

Returns Does not return anything but makes a plot.

Author: SMM

LSDPlottingTools.LSDMap_ChiPlotting.**StackedProfilesGradient**(*chi_csv_fname*, *FigFileName=u'Image.pdf'*, *FigFormat=u'show'*, *elevation_threshold=0*, *first_basin=0*, *last_basin=0*, *basin_order_list=[]*, *basin_rename_list=[]*, *this_cmap=<Mock name='mock.cm.cubehelix' id='139783202333904'>*, *data_name=u'chi'*, *X_offset=5*, *plotting_data_format=u'log'*, *label_sources=False*, *source_thinning_threshold=0*, *size_format=u'ESURF'*)

This function plots the chi vs elevation or flow distance vs elevation.

It stacks profiles (so the basins are spaced out). It colours the plots by the chi steepness (which is equal to the normalised channel steepness if A_0 is set to 1).

Parameters

- **chi_csv_fname** (*str*) – The name (with full path and extension) of the cdv file with chi, chi slope, etc information. This file is produced by the chi_mapping_tool.
- **FigFileName** (*str*) – The name of the figure file
- **FigFormat** (*str*) – The format of the figure. Usually 'png' or 'pdf'. If "show" then it calls the matplotlib show() command.
- **elevation_threshold** (*float*) – elevation_threshold chi points below this elevation are removed from plotting.
- **first_basin** (*int*) – The basin to start with (but overridden by the basin list)
- **last_basin** (*int*) – The basin to end with (but overridden by the basin list)
- **basin_order_list** (*int list*) – The basins to plot
- **basin_rename_list** (*int list*) – A list for naming substitutions. Useful because LSDTopoTools might number basins in a way a human wouldn't, so a user can intervene in the names.

- **this_cmap** (*colormap*) – NOT USED! We now use a default colourmap but this may change.
- **data_name** (*str*) – 'chi' or 'flow_distance' What to plot along the x-axis.
- **X_offset** (*float*) – The offest in chi between the basins along the x-axis. Used to space out the profiles so you can see each of them.
- **plotting_data_format** – NOT USED previously if 'log' use logarithm scale, but we now automatically do this. Might change later.
- **label_sources** (*bool*) – If true, label the sources.
- **source_thinning_threshold** (*float*) –
- **size_format** (*str*) – Can be "big" (16 inches wide), "geomorphology" (6.25 inches wide), or "ESURF" (4.92 inches wide) (defualt esurf).

Returns Does not return anything but makes a plot.

Author: SMM

LSDPlottingTools.LSDMap_GDALIO module

LSDPlottingTools.LSDMap_GDALIO.**CheckNoData**(*FileName*)

This looks through the head file of an ENVI raster and if it doesn't find the nodata line it rewrites the file to include the nodata line.

Parameters **FileName** (*str*) – The filename (with path and extension) of the raster.

Returns The total number of pixels (although what it is really doing is updating the header file. The return is just to check if it is working and yes I know this is stupid.)

Return type int

Author: SMM

LSDPlottingTools.LSDMap_GDALIO.**GetGeoInfo**(*FileName*)

This gets information from the raster file using gdal

Parameters **FileName** (*str*) – The filename (with path and extension) of the raster.

Returns

A vector that contains:

- NDV: the nodata values
- xsize: cellsize in x direction
- ysize: cellsize in y direction
- GeoT: the tranform (a string)
- Projection: the Projection (a string)
- DataType: The type of data (an int explaing the bits of each data element)

Return type float

Author: SMM

LSDPlottingTools.LSDMap_GDALIO.**GetLocationVectors**(*FileName*)
This gets a vector of the x and y locations of the coordinates

Note: This assumes raster is already projected into UTM, and is in ENVI format! It reads from an ENVI header file.

Parameters **FileName** (*str*) – The filename (with path and extension) of the raster.

Returns A vector of the x locations (eastings) float: A vector of the y locations (northings)

Return type float

Author: SMM

LSDPlottingTools.LSDMap_GDALIO.**GetNPixelsInRaster**(*FileName*)
This gets the total number of pixels in the raster

Parameters **FileName** (*str*) – The filename (with path and extension) of the raster.

Returns The total number of pixels

Return type int

Author: SMM

LSDPlottingTools.LSDMap_GDALIO.**GetPixelArea**(*FileName*)
Gets the area in m^2 of the pixels

Parameters **rasterfn** (*str*) – The filename (with path and extension) of the raster

Returns Pixel_area (float): The area of each pixel

Return type float

Author: SMM

LSDPlottingTools.LSDMap_GDALIO.**GetRasterExtent**(*FileName*)
This gets a vector of the minimums and maximums of the coordinates

Note: This assumes raster is already projected into UTM, and is in ENVI format! It reads from an ENVI header file.

Parameters **FileName** (*str*) – The filename (with path and extension) of the raster.

Returns

A vector that contains

- extent[0]: XMin
- extent[1]: XMax
- extent[2]: YMin
- extent[3]: YMax

Return type float

Author: SMM

LSDPlottingTools.LSDMap_GDALIO.**GetUTMEPSG**(*FileName*)
Uses GDAL to get the EPSG string from the raster.

Parameters **FileName** (*str*) – The filename (with path and extension) of the raster.

Returns The EPSG string

Return type str

Author: SMM

LSDPlottingTools.LSDMap_GDALIO.**GetUTMMaxMin**(*FileName*)
This gets the minimum and maximum UTM values.

WARNING it assumes raster is already projected into UTM, and is in ENVI format! It reads from an ENVI header file.

Parameters **FileName** (*str*) – The filename (with path and extension) of the raster

Returns The cell size in metres float: The X minimum (easting) in metres float: The X maximum (easting) in metres float: The Y minimum (northing) in metres float: The Y maximum (northing) in metres

Return type float

Author: SMM

LSDPlottingTools.LSDMap_GDALIO.**GetUTMMaxMinFromRowsCol**(*FileName*, *x_max_col*, *x_min_col*, *y_max_row*, *y_min_row*)
This gets the minimum and maximum UTM values but you give it the row and column numbers.

Note: This assumes raster is already projected into UTM, and is in ENVI format! It reads from an ENVI header file.

Parameters

- **FileName** (*str*) – The filename (with path and extension) of the raster
- **x_max_col** (*int*) – The column to use as the maximum
- **x_min_col** (*int*) – The column to use as the minimum
- **y_max_row** (*int*) – The row to use as the maximum
- **y_min_row** (*int*) – The row to use as the minimum

Returns The X maximum (easting) in metres float: The X minimum (easting) in metres float: The Y maximum (northing) in metres float: The Y minimum (northing) in metres

Return type float

Author: SMM

LSDPlottingTools.LSDMap_GDALIO.**RasterDifference**(*RasterFile1*, *RasterFile2*, *raster_band=1*, *OutFileName=u'Test.outfile'*, *OutFileType=u'ENVI'*)
Takes two rasters of same size and subtracts second from first, e.g. Raster1 - Raster2 = raster_of_difference then writes it out to file

LSDPlottingTools.LSDMap_GDALIO.**ReadRasterArrayBlocks**(*raster_file*, *raster_band=1*)
This reads a raster file (from GDAL) into an array. The "blocks" bit makes it efficient. :param FileName: The filename (with path and extension) of the raster. :type FileName: str :param raster_band: the band of the raster (almost all uses with LSDTopoTools will have a 1 band raster) :type raster_band: int

Returns A numpy array with the data from the raster.

Return type np.array

Author: SMM

LSDPlottingTools.LSDMap_GDALIO.**array2raster**(*rasterfn*, *newRasterfn*, *array*, *driver_name=u'ENVI'*, *noDataValue=-9999*)
Takes an array and writes to a GDAL compatible raster. It needs another raster to map the dimensions.

Parameters

- **FileName** (*str*) – The filename (with path and extension) of a raster that has the same dimensions as the raster to be written.
- **newRasterfn** (*str*) – The filename (with path and extension) of the new raster.
- **array** (*np.array*) – The array to be written
- **driver_name** (*str*) – The type of raster to write. Default is ENVI since that is the LSDTOpoTools format
- **noDataValue** (*float*) – The no data value

Returns A numpy array with the data from the raster.

Return type np.array

Author: SMM

LSDPlottingTools.LSDMap_GDALIO.**getNoDataValue**(*rasterfn*)
This gets the nodata value from the raster

Parameters **rasterfn** (*str*) – The filename (with path and extension) of the raster

Returns nodatavalue; the nodata value

Return type float

Author: SMM

LSDPlottingTools.LSDMap_GDALIO.**setNoDataValue**(*rasterfn*)
This sets the nodata value from the raster

Parameters **rasterfn** (*str*) – The filename (with path and extension) of the raster

Returns None

Author: SMM

LSDPlottingTools.LSDMap_PointTools module

LSDPlottingTools.LSDMap_PointTools.**ConvertAllCSVToGeoJSON**(*path*)
This looks in a directory and converts all .csv files to GeoJSON.

This is handy if, for example, you want to display data on the web using leaflet or D3.js

Note: This assumes your csv files have latitude and longitude columns. If the LSDMap_PointData object will not be able to read them.

Parameters **path** (*str*) – The path in which you want to convert the csv files

Returns None, but you will get a load of GeoJSON files.

Author: SMM

LSDPlottingTools.LSDMap_PointTools.**ConvertAllCSVToShapefile**(*path*)
This looks in a directory and converts all .csv files to shapefiles

This is handy if, for example, you want to display data using ArcMap of QGIS

Note: This assumes your csv files have latitude and longitude columns. If the LSDMap_PointData object will not be able to read them.

Parameters **path** (*str*) – The path in which you want to convert the csv files

Returns None, but you will get a load of GeoJSON files.

Author: SMM

class LSDPlottingTools.LSDMap_PointTools.**LSDMap_PointData**(*FileName*)
Bases: object

GetLatitude(*PrintToScreen=False*)
Gets the latitude list.

Parameters **PrintToScreen** (*bool*) – If true, prints to screen

Returns A list of the latitudes

Return type float

Author: SMM

GetLongitude(*PrintToScreen=False*)
Gets the longitude list.

Parameters **PrintToScreen** (*bool*) – If true, prints to screen

Returns A list of the longitudes

Return type float

Author: SMM

GetParameterNames(*PrintToScreen=False*)
Gets the list of parameter names.

Parameters **PrintToScreen** (*bool*) – If true, prints to screen

Returns A list of the variable names

Return type str

Author: SMM

GetParameterTypes(*PrintToScreen=False*)
Gets the tyes of each names.

Parameters **PrintToScreen** (*bool*) – If true, prints to screen

Returns A list of the variable types

Return type str

Author: SMM

GetUTMEastingNorthing(*EPSG_string*)
Returns two lists: the latitude and longitude converted to northing and easting.

Parameters

- **PrintToScreen** (*bool*) – If true, prints to screen.
- **EPSG_string** (*str*) – The EPSG code of the UTM coordinates you want (326XX) with zone XX is for north, 327XX is for south.

Returns Two lists containing easting and northing

Return type float

Author: SMM

GetUTMEastingNorthingFromQuery(*EPSG_string*, *Latitude_string*, *Longitude_string*)
Returns two lists: the latitude and longitude converted to northing and easting. But you can define the columns if there are more than one latitude and longitude columns.

Note: This is used mainly if there are multple lat-long coordinates in the csv file. For example when you have basin centroids and basin outlets in the same file.

Parameters

- **PrintToScreen** (*bool*) – If true, prints to screen.
- **EPSG_string** (*str*) – The EPSG code of the UTM coordinates you want (326XX) with zone XX is for north, 327XX is for south.
- **Latitude_string** (*str*) – The name of the latitude column you want
- **Longitude_string** (*str*) – The name of the longitude column you want.

Returns Two lists containing easting and northing

Return type float

Author: SMM

QueryData(*data_name*, *PrintToScreen=False*)
Returns the list of the data that has the column header data_name

Parameters

- **PrintToScreen** (*bool*) – If true, prints to screen.
- **data_name** (*str*) – The header of the column you want

Returns A list of the data

Return type float

Author: SMM

ThinData(*data_name*, *Threshold_value*)
This removes data from a point function that is below a threshold value

Parameters

- **data_name** (*str*) – The name of the data member to select
- **Threshold_value** (*float*) – Below this threshold points will be removed.

Returns None removes data from the object (not reversible!!)

Author: SMM

ThinDataSelection(*data_name*, *data_for_selection_list*)
This function takes a list of values and retains the members in data name corresponding to that selection

Parameters

- **data_name** (*str*) – The name of the data member to select
- **data_for_selection_list** (*int*) – A list of values to retain. Useful for things like selecting basins or sources.

Returns None removes data from the object (not reversible!!)

Author: SMM

TranslateToReducedGeoJSON(*FileName*)
This converts the point data to a GeoJSON

Parameters **FileName** (*str*) – the name of the file to be printed. The code strips the extension and turns it into .geojson, so you can give it the name of the csv file and ti will still work.

Returns None, but prints a new GeoJSON

Author: SMM

TranslateToReducedShapefile(*FileName*)
This converts the point data to a shapefile

Parameters **FileName** (*str*) – the name of the file to be printed. The code strips the extension and turns it into .shp, so you can give it the name of the csv file and ti will still work.

Returns None, but prints a new shapefile

Author: SMM

LSDPlottingTools.LSDMap_Subplots module

LSDPlottingTools.LSDMap_Subplots.**MultiDrapeErodeDiffMaps**(*DataDir*, *ElevationRaster*, *DrapeRasterWild*, *cmap*, *drape_min_threshold=None*, *cbar_label=None*, *drape_max_threshold=None*, *middle_mask_range=None*)

Plots multiple drape maps of erosion/deposition (a DEM of difference) over a hillshade raster of the basin.

Takes a wildcard for the drapes Expexts a single elevation raster for the background hillshade, but this could be modified in future.

Parameters

- **DataDir** (*str*) – Path to the directory containing the data files

- **ElevationRaster** (*str*) – Name of the elevation raster used to create the hillshade
- **DrapeRasterWild** (*str*) – Wildcard string used to find all the drape files in the directory.
- **cmap** – Can be the string name of a colourmap, or a Colourmap object
- **drape_min_threshold** (*float, optional*) – Minimum value for the drape raster, i.e. values below this threshold will be masked and not plotted.
- **drape_max_threshold** (*float, optional*) – Maximum value for the drape raster, i.e. values above this value will be masked and not plotted.
- **cbar_label** – Label for the colourbar on the figure. This is the colourbar for the drape colourmap.

Notes

Consider, if plotting multiple datasets, how you are going to deal with min a max values in the colur range. imshow will automatically set vmin and vmax and stretch the colour bar over this - which can be visually misleading. Ideally, you want to have the same colour map used for *all* subplots, and this is not default behaviour.

Note: If *drape_max_threshold* is not set, the function searches for the maximum value in the range of rasters found by expanding the *DrapeRasterWild* argument and searching for the maximum value out of all rasters found.

Raises `Exception` – If the maximum value in the drape maps could not be found.

Author: DAV & FJC

`LSDPlottingTools.LSDMap_Subplots.`**`MultiDrapeFloodMaps`**(*DataDir*, *ElevationRaster*, *DrapeRasterWild*, *cmap*, *drape_min_threshold=None*, *drape_max=None*, *cbar_label=None*)

Creates a figure with multiple drape maps over a hillshade.

Plots flood extents from water depth rasters draped over the catchment elevation raster in a series of subplots

Takes a wildcard for the drapes Expexts a fixed elevation raster, but this could be modified in future.

Parameters

- **DataDir** (*str*) – Path to the directory containing the data files
- **ElevationRaster** (*str*) – Name of the elevation raster used to create the hillshade
- **DrapeRasterWild** (*str*) – Wildcard string used to find all the drape files in the directory.
- **cmap** – Can be the string name of a colourmap, or a Colourmap object
- **drape_min** (*float, optional*) – Minimum value for the drape raster, i.e. values below this threshold will be masked and not plotted.
- **drape_max** (*float, optional*) – Maximum value for the drape raster, i.e. values above this value will be masked and not plotted.
- **cbar_label** (*str, optional*) – Label for the colourbar on the figure. This is the colourbar for the drape colourmap.

Notes

Consider, if plotting multiple datasets, how you are going to deal with min a max values in the colur range. imshow will automatically set vmin and vmax and stretch the colour bar over this - which can be visually misleading. Ideally, you want to have the same colour map used for *all* subplots, and this is not default behaviour.

Note: If *drape_max* is not set, the function searches for the maximum value in the range of rasters found by expanding the *DrapeRasterWild* argument and searching for the maximum value out of all rasters found.

Raises `Exception` – If the maximum value in the drape maps could not be found.

`LSDPlottingTools.LSDMap_Subplots.`**`cm2inch`**(*value*)
Convert cm to inch for figure sizing

`LSDPlottingTools.LSDMap_Subplots.`**`field_sites`**(*DataDirectory*, *N_HSFiles*, *NRows*, *NCols*, *n_target_ticks*)

`LSDPlottingTools.LSDMap_Subplots.`**`findmaxval_multirasters`**(*FileList*)
Loops through a list or array of rasters (np arrays) and finds the maximum single value in the set of arrays.

`LSDPlottingTools.LSDMap_Subplots.`**`findminval_multirasters`**(*FileList*)
Loops through a list or array of rasters (np arrays) and finds the minimum single value in the set of arrays.

`LSDPlottingTools.LSDMap_Subplots.`**`flood_maps_with_shapefile`**(*DataDirectory*)

`LSDPlottingTools.LSDMap_Subplots.`**`multiple_flood_maps`**(*DataDirectory*)
Make nice subplots of floodplain rasters for different field sites

LSDPlottingTools.adjust_text module

`LSDPlottingTools.adjust_text.`**`adjust_text`**(*texts*, *x=None*, *y=None*, *add_objects=None*, *ax=None*, *expand_text=(1.2, 1.2)*, *expand_points=(1.2, 1.2)*, *expand_objects=(1.2, 1.2)*, *expand_align=(0.9, 0.9)*, *autoalign=u'xy'*, *va=u'center'*, *ha=u'center'*, *force_text=0.5*, *force_points=0.5*, *force_objects=0.5*, *lim=100*, *precision=0*, *only_move={}*, *text_from_text=True*, *text_from_points=True*, *save_steps=False*, *save_prefix=u''*, *save_format=u'png'*, *add_step_numbers=True*, *draggable=True*, **args*, ***kwargs*)
Iteratively adjusts the locations of texts. First moves all texts that are outside the axes limits inside. Then in each iteration moves all texts away from each other and from points. In the end hides texts and substitutes them with annotations to link them to the respective points.

Parameters

- **`texts`** (*`list`*) – a list of text.Text objects to adjust
- **`x`** (*`seq`*) – x-coordinates of points to repel from; if not provided only uses text coordinates
- **`y`** (*`seq`*) – y-coordinates of points to repel from; if not provided only uses text coordinates
- **`add_objects`** (*`list`*) – a list of additional matplotlib objects to avoid; they must have a .get_window_extent() method
- **`ax`** (*`obj`*) – axes object with the plot; if not provided is determined by plt.gca()

- **expand_text** (*seq*) – a tuple/list/... with 2 numbers (x, y) to expand texts when repelling them from each other; default (1.2, 1.2)
- **expand_points** (*seq*) – a tuple/list/... with 2 numbers (x, y) to expand texts when repelling them from points; default (1.2, 1.2)
- **expand_objects** (*seq*) – a tuple/list/... with 2 numbers (x, y) to expand texts when repelling them from points; default (1.2, 1.2)
- **expand_align** (*seq*) – a tuple/list/... with 2 numbers (x, y) to expand texts when autoaligning texts; default (1., 1.)
- **autoalign** – If 'xy', the best alignment of all texts will be determined in all directions automatically before running the iterative adjustment; if 'x' will only align horizontally, if 'y' - vertically; overrides va and ha
- **va** (*str*) – vertical alignment of texts
- **ha** (*str*) – horizontal alignment of texts
- **force_text** (*float*) – the repel force from texts is multiplied by this value; default 0.5
- **force_points** (*float*) – the repel force from points is multiplied by this value; default 0.5
- **force_objects** (*float*) – same as other forces, but for repelling additional objects
- **lim** (*int*) – limit of number of iterations
- **precision** (*float*) – up to which sum of all overlaps along both x and y to iterate; may need to increase for complicated situations; default 0, so no overlaps with anything.
- **only_move** (*dict*) – a dict to restrict movement of texts to only certain axis. Valid keys are 'points' and 'text', for each of them valid values are 'x', 'y' and 'xy'. This way you can forbid moving texts along either of the axes due to overlaps with points, but let it happen if there is an overlap with texts: only_move={'points':'y', 'text':'xy'}. Default: None, so everything is allowed.
- **text_from_text** (*bool*) – whether to repel texts from each other; default True
- **text_from_points** (*bool*) – whether to repel texts from points; default True; can helpful to switch of in extremely crouded plots
- **save_steps** (*bool*) – whether to save intermediate steps as images; default False
- **save_prefix** (*str*) – a path and/or prefix to the saved steps; default ''
- **save_format** (*str*) – a format to save the steps into; default 'png
- **and **kwargs** (**args*) – any arguments will be fed into plt.annotate after all the optimization is done just for plotting
- **add_step_numbers** (*bool*) – whether to add step numbers as titles to the images of saving steps
- **draggable** (*bool*) – whether to make the annotations draggable; default True

LSDPlottingTools.adjust_text.**get_bboxes**(*objs*, *r*, *expand=(1.0*, *1.0)*, *ax=None*)

LSDPlottingTools.adjust_text.**get_midpoint**(*bbox*)

LSDPlottingTools.adjust_text.**get_points_inside_bbox**(*x*, *y*, *bbox*)

LSDPlottingTools.adjust_text.**get_renderer**(*fig*)

LSDPlottingTools.adjust_text.**move_texts**(*texts*, *delta_x*, *delta_y*, *bboxes=None*, *renderer=None*, *ax=None*)

LSDPlottingTools.adjust_text.**optimally_align_text**(*x*, *y*, *texts*, *expand=(1.0, 1.0)*, *add_bboxes=[]*, *renderer=None*, *ax=None*, *direction=u'xy'*)

For all text objects find alignment that causes the least overlap with points and other texts and apply it

LSDPlottingTools.adjust_text.**overlap_bbox_and_point**(*bbox*, *xp*, *yp*)

LSDPlottingTools.adjust_text.**repel_text**(*texts*, *renderer=None*, *ax=None*, *expand=(1.2, 1.2)*, *only_use_max_min=False*, *move=False*)

Repel texts from each other while expanding their bounding boxes by expand (x, y), e.g. (1.2, 1.2) would multiply width and height by 1.2. Requires a renderer to get the actual sizes of the text, and to that end either one needs to be directly provided, or the axes have to be specified, and the renderer is then got from the axes object.

LSDPlottingTools.adjust_text.**repel_text_from_axes**(*texts*, *ax=None*, *bboxes=None*, *renderer=None*, *expand=None*)

LSDPlottingTools.adjust_text.**repel_text_from_bboxes**(*add_bboxes*, *texts*, *renderer=None*, *ax=None*, *expand=(1.2, 1.2)*, *only_use_max_min=False*, *move=False*)

Repel texts from other objects' bboxes while expanding their (texts') bounding boxes by expand (x, y), e.g. (1.2, 1.2) would multiply width and height by 1.2. Requires a renderer to get the actual sizes of the text, and to that end either one needs to be directly provided, or the axes have to be specified, and the renderer is then got from the axes object.

LSDPlottingTools.adjust_text.**repel_text_from_points**(*x*, *y*, *texts*, *renderer=None*, *ax=None*, *expand=(1.2, 1.2)*, *move=False*)

Repel texts from all points specified by x and y while expanding their (texts'!) bounding boxes by expandby (x, y), e.g. (1.2, 1.2) would multiply both width and height by 1.2. In the case when the text overlaps a point, but there is no definite direction for movement, moves in random direction by 40% of it's width and/or height depending on Requires a renderer to get the actual sizes of the text, and to that end either one needs to be directly provided, or the axes have to be specified, and the renderer is then got from the axes object.

LSDPlottingTools.cubehelix module

LSDPlottingTools.cubehelix.**cmap**(*start=0.5*, *rot=-1.5*, *gamma=1.0*, *reverse=False*, *nlev=256.0*, *minSat=1.2*, *maxSat=1.2*, *minLight=0.0*, *maxLight=1.0*, ***kwargs*)

A full implementation of Dave Green's "cubehelix" for Matplotlib. Based on the FORTRAN 77 code provided in D.A. Green, 2011, BASI, 39, 289.

http://adsabs.harvard.edu/abs/2011arXiv1108.5083G

User can adjust all parameters of the cubehelix algorithm. This enables much greater flexibility in choosing color maps, while always ensuring the color map scales in intensity from black to white. A few simple examples:

Default color map settings produce the standard "cubehelix".

Create color map in only blues by setting rot=0 and start=0.

Create reverse (white to black) backwards through the rainbow once by setting rot=1 and reverse=True.

Parameters

- **start** (*scalar, optional*) – Sets the starting position in the color space. 0=blue, 1=red, 2=green. Defaults to 0.5.

- **rot** (*scalar, optional*) – The number of rotations through the rainbow. Can be positive or negative, indicating direction of rainbow. Negative values correspond to Blue->Red direction. Defaults to -1.5
- **gamma** (*scalar, optional*) – The gamma correction for intensity. Defaults to 1.0
- **reverse** (*boolean, optional*) – Set to True to reverse the color map. Will go from black to white. Good for density plots where shade~density. Defaults to False
- **nlev** (*scalar, optional*) – Defines the number of discrete levels to render colors at. Defaults to 256.
- **sat** (*scalar, optional*) – The saturation intensity factor. Defaults to 1.2 NOTE: this was formerly known as "hue" parameter
- **minSat** (*scalar, optional*) – Sets the minimum-level saturation. Defaults to 1.2
- **maxSat** (*scalar, optional*) – Sets the maximum-level saturation. Defaults to 1.2
- **startHue** (*scalar, optional*) – Sets the starting color, ranging from [0, 360], as in D3 version by @mbostock NOTE: overrides values in start parameter
- **endHue** (*scalar, optional*) – Sets the ending color, ranging from [0, 360], as in D3 version by @mbostock NOTE: overrides values in rot parameter
- **minLight** (*scalar, optional*) – Sets the minimum lightness value. Defaults to 0.
- **maxLight** (*scalar, optional*) – Sets the maximum lightness value. Defaults to 1.

Returns

Return type matplotlib.colors.LinearSegmentedColormap object

Example

```
>>> import cubehelix
>>> import matplotlib.pyplot as plt
>>> import numpy as np
>>> x = np.random.randn(1000)
>>> y = np.random.randn(1000)
>>> cx = cubehelix.cmap(start=0., rot=-0.5)
>>> plt.hexbin(x, y, gridsize=50, cmap=cx)
```

2014-04 (@jradavenport) Ported from IDL version 2014-04 (@jradavenport) Added kwargs to enable similar to D3 version,

changed name of "hue" parameter to "sat"

LSDPlottingTools.colours module

A set of functions to perform modifications to matplotlib colourbars, and colourmaps, beyond the core maplotlib colourbar/colourmap functionality.

Created on Thu Jan 12 14:33:21 2017

Author: DAV

class LSDPlottingTools.colours.**MetaColours**
Bases: type

A metclass for colourmaps making them all read-only attributes

darkearth
A terrain colour map that doesn't have the stupid blue colour for low lying land...

niceterrain
A terrain colour map that doesn't have the stupid blue colour for low lying land...

class LSDPlottingTools.colours.**TransformedColourmap**(*function*, *cmap*)
Bases: object

Applies function (which should operate on vectors of shape 3: [r, g, b]), on colormap cmap. This routine will break any discontinuous points in a colormap.

DOES NOT WORK!

class LSDPlottingTools.colours.**UsefulColourmaps**
Bases: object

The interface for accessing usefulcolourmaps attributes

LSDPlottingTools.colours.**cmap_discretize**(*N*, *cmap*)
Return a discrete colormap from the continuous colormap cmap.

Parameters

- **cmap** – colormap instance, eg. cm.jet.
- **N** – number of colors.

Example

x = resize(arange(100), (5,100)) djet = cmap_discretize(cm.jet, 5) imshow(x, cmap=djet)

LSDPlottingTools.colours.**colorbar_index**(*fig*, *cax*, *ncolors*, *cmap*, *drape_min_threshold*, *drape_max*)

State-machine like function that creates a discrete colormap and plots it on a figure that is passed as an argument.

Parameters

- **fig** (*matplotlib.Figure*) – Instance of a matplotlib figure object.
- **cax** (*matplotlib.Axes*) – Axes instance to create the colourbar from. This must be the Axes containing the data that your colourbar will be mapped from.
- **ncolors** (*int*) – The number of colours in the discrete colourbar map.
- **cmap** (*str or Colormap object*) – Either the name of a matplotlib colormap, or an object instance of the colormap, e.g. cm.jet
- **drape_min_threshold** (*float*) – Number setting the threshold level of the drape raster This should match any threshold you have set to mask the drape/overlay raster.
- **drape_max** (*float*) – Similar to above, but for the upper threshold of your drape mask.

LSDPlottingTools.colours.**discrete_colourmap**(*N*, *base_cmap=None*)
Creates an N-bin discrete colourmap from the specified input colormap.

Author: github.com/jakevdp adopted by DAV

Note: Modified so you can pass in the string name of a colourmap or a Colormap object.

Parameters

- **N** (*int*) – Number of bins for the discrete colourmap. I.e. the number of colours you will get.
- **base_cmap** (*str or Colormap object*) – Can either be the name of a colourmap e.g. "jet" or a matplotlib Colormap object

LSDPlottingTools.colours.**nonlinear_colormap**()
Creates a non-linear colourmap from an existing colourmap.

LSDPlottingTools.colours.**truncate_colormap**(*cmap*, *minval=0.0*, *maxval=1.0*, *n=-1*)
Truncates a standard matplotlib colourmap so that you can use part of the colour range in your plots. Handy when the colourmap you like has very light values at one end of the map that can't be seen easily.

Parameters

- **(** (cmap) – obj: *Colormap*): A matplotlib Colormap object. Note this is not a string name of the colourmap, you must pass the object type.
- **minval** (*int, optional*) – The lower value to truncate the colour map to. colourmaps range from 0.0 to 1.0. Should be 0.0 to include the full lower end of the colour spectrum.
- **maxval** (*int, optional*) – The upper value to truncate the colour map to. maximum should be 1.0 to include the full upper range of colours.
- **n** (*int*) – Leave at default.

Example

minColor = 0.00 maxColor = 0.85 inferno_t = truncate_colormap(_plt.get_cmap("inferno"), minColor, maxColor)

LSDPlottingTools.labels module

Created on Mon Oct 10 16:09:29 2016

A series of functions to provide extra functionality to matplotlib involving the creation of labels for plots.

Author: DAV

@stackoverflow: http://stackoverflow.com/questions/16992038/inline-labels-in-matplotlib

LSDPlottingTools.labels.**labelLine**(*line*, *x*, *label=None*, *align=True*, ***kwargs*)
Places a label on a line plot and orients it to run parallel with the line.

Given a matplotlib Line instance and x-coordinate, places *label* at the x-coord on the given line and orientates it parallel to the line.

Author: http://stackoverflow.com/questions/16992038/inline-labels-in-matplotlib

Parameters

- **line** – Matplotlib Line instance
- **x** – x-coordinate on the line at which the label will be positioned.
- **label** (*str*) – The label to be added to the line.
- **align** (*bool*) – whether or not to align the label parallel to the line

LSDPlottingTools.labels.**labelLines**(*lines*, *align=True*, *xvals=None*, ***kwargs*)

Version of labelLine that assigns labels for all lines in a plot.

Similar to labelLine, except a list of lines is passed.

Argumnets: lines (list): A list of the lines to be labeled. xvals: A list of x-coordinates where the labels should be anchored.

LSDPlottingTools.labels.**make_line_label**(*fname*)

Makes a string (label) by splitting a file name.

> **Warning:** A lot of this is hard coded to split according to certain filenaming conventions, separated by underscored. e.g. MyFile_part1_part2_part3.file So you should modify this to fit your own file naming convention.

Todo: Rewrite this as a more generic function.

Parameters **fname** (*str*) – Filename to create labels from.

Author: DAV

LSDPlottingTools.locationmap module

Location map.

Plots a location map using the Cartopy package

Install cartopy first for this to work.

http://scitools.org.uk/cartopy/docs/v0.13/index.html

Add annotations for locations using their lon/lats.

Author: DAV

LSDPlottingTools.locationmap.**location_map**(*extent*, *gazetter*, *offset=0.0*)

Plots a series of points marking towns/sample sites/locations etc given a dictionary of places and lat/lons.

Parameters

- **extent** (*list*) – A list of the cooridinates of the bounding extent of the location map in format: [West, East, South, North] e.g.:

  [lonW, lonE, latS, latN]

- **gazetter** (*dict*) – A dictionary of arbitrary length of the format: { 'Placename1' : (LATITUDE, LONGITUDE),

  'Placename2' : (LATITUDE, LONGITUDE). and so on...}

- **offset** (*float*) – Offset of the text label to the marker points, in degrees.

Todo: Greying out the landmass appears to block out the marker points (weird...) so this has been commented out and you just get an outline map for now.

Author: DAV

LSDPlottingTools.scalebar module

A set of functions to create scalebars and other miscellany for maps.

Created on Thu Jan 12 14:33:21 2017

Author: Adapted from Philippe Pinard's excellent scalebar module to be integrated with LSDMappingTools (DAV)

https://github.com/ppinard/matplotlib-scalebar

Example:

```
>>> fig = plt.figure()
>>> ax = fig.add_axes([0.0, 0.0, 1.0, 1.0])
>>> ax.imshow(...)
>>> scalebar = ScaleBar(0.2)
>>> ax.add_artist(scalebar)
>>> plt.show()
```

Module contents

Created on Fri Oct 30 10:37:16 2015

@author: smudd

automodule:: LSDMappingTools

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

l

Index

G

H

I

L

M

N