Manual for Package: gis
Revision 1

Karl Kästner

March 28, 2020

Contents

1 gis

1.1 GPX

1.2 batavia_zero

2 centreline/@Centreline

2.1 Centreline

2.2 channel_planimetry

2.3 clip

2.4 connect_graph

2.5 curvature

2.6 cut

2.7 determine_width

2.8 distance

2.9 export_cross_section

2.10 export_node

2.11 export_shp

2.12 find_nearest_segment

2.13 from_polygon

2.14 from_shp

2.15 get

2.16 init

```
obj.seg_S(id(end)) = NaN;
```

2.17 init_connect

2.18 init_node_D

2.19 link_centreline

2.20 plot

2.21 plot_connection

2.22 prune

2.23 prune_leaves

2.24 prune_manually

2.25 reachable

2.26 remove_duplicate_points

2.27 resample

2.28 routing

2.29 routing2

2.30 shp_resample_simple

2.31 snmesh

4.2 thalweg

4.3 xy2sn_quadratic

5 gis

5.1 gpx_export_csv

5.2 hgt_plot

5.3 hgt_read

```
% [ floor(mednan(z(kk))) meannan(z(kk)) min(z(kk)) max(z(kk)) ]
```

5.4 hgt_read_all

5.5 hgt_resample

5.6 nmeatime

6 shapefile/@Shp

6.1 Shp

6.2 area

6.3 buffer

6.4 cat

6.5 clip

6.6 clip_rect

6.7 close_polygon

6.8 concat

6.9 connect_network

```
 TODO make unique
 attach segments to
XY = [cvec(shp.X),shp.;
 knnsearch for nearest n neighbours
 for each segment
```

6.10 contour

6.11 cp

6.12 create

6.13 curvature

6.14 cut

6.15 diameter

6.16 edges

6.17 export_geo

6.18 export_gpx

6.19 export_gpx_track

6.20 export_ldb

6.21 export_poly

6.22 export_sdf

6.23 export_spline

6.24 extract_coastline

6.25 first_point

6.26 flat

6.27 generate_four_colour_index

6.28 import_geo

6.29 import_poly

6.30 join_lines

6.31 last_point

6.32 length

6.33 length2

6.34 line2point

6.35 link_lines

6.36 make_clockwise

6.37 merge

6.38 merge2

6.39 padd_nan

6.40 plot

6.41 points

6.42 polygon_boundary

6.43 read

6.44 readZ

6.45 remove_duplicate_points

6.46 remove_leaves

6.47 remove_nan

6.48 remove_polygon_closure

6.49 remove_short_elements

6.50 renumber

6.51 resample

6.52 resample_2

6.53 resample_min

6.54 resample_quick

6.55 scale

6.56 segment

6.57 select_for_refinement

6.58 set_geometry

6.59 set_resolution

6.60 skip

6.61 smooth

6.62 split_jump

6.63 split_line

6.64 split_nan

6.65 swap_hemisphere

6.66 translate

6.67 write

7 shapefile

7.1 astar_multi

7.2 astar_recursive

```
astar path finding algorithm
```

7.3 edge_chain

7.4 edge_from_bnd

7.5 preload_shp

7.6 read_gpx

7.7 shapewrite__

```
Copyright (C) 2014,2015 Philip Nienhuis

This program is free software; you can redistribute it and/or
    modify it
under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program. If not, see <http://www.gnu.org/licenses
    />.
-*- texinfo -*-
@deftypefn {Function File} {@var{status} =} shapewrite (@var{shpstr
    }, @var{fname})
Write contents of map- or geostruct to a GIS shape file.

@var{shpstr} must be a valid mapstruct or geostruct, a struct array
     with an
entry for each shape feature, with fields Geometry, BoundingBox,
    and X and Y
```

```
(mapstruct) or Lat and Lon (geostruct). For geostructs, Lat and Lon
     field
data will be written as X and Y data. Field Geometry can have data
    values
of 'Point', 'MultiPoint', 'Line', or 'Polygon', all case-
    insensitive. For
each shape feature, field BoundingBox should contain the minimum
    and maximum
(X,Y) coordinates in a 2x2 array [minX, minY; maxX, maxY]. The X
    and Y
fields should contain X (or Latitude) and Y (or Longitude)
    coordinates for
each point or vertex as row vectors; for polylines and polygons
    vertices of
each subfeature (if present) should be separated by NaN entries.

@var{fname} should be a valid shape file name, optionally with a '.
    shp'
suffix.

shapewrite produces 2 or 3 files, i.e. a .shp file (the actual
    shape file),
a .shx file (index file), and if @var{shpstr} contained additional
    fields,
a .dbf file (dBase type 3) with the contents of those additional
    fields.

@var{status} is 1 if the shape file set was written successfully, 0
otherwise.

@seealso{shaperead, shapeinfo}
@end deftypefn
Author: Philip Nienhuis <prnienhuis@users.sf.net>
Created: 2014-12-30
Input validation
Assess shape variable type (oct or ml/geo ml/map)
Yep. Find out what type
Assume it is an Octave-style struct read by shaperead
Assume it is a Matlab-style mapstruct
Assume it is a Matlab-style geostruct
Not a supported struct type
Check file name
Later on bname.shx and bname.dbf will be read
Prepare a few things
Change Lat/Lon fields into X/Y
Only now (after input checks) open .shp and .shx files & rewind
    just to be sure
Write headers in .shp & .shx (identical). First magic number 9994 +
     5 zeros
```

```
In between here = filelength in 16-bit words (single). For .shx it'
    s known
Next, shp file version
Shape feature type
Bounding box. Can be run later for ML type shape structs. Fill with
     zeros
Skip to start of first record position
Write shape features one by one
Write record start pos to .shx file
Write record contents
Point
Record index number
Record length (fixed)
Shape type
Simply write XY cordinates
MultiPoint
Record index number
Record length
Shape type
Bounding box
Nr of points
Polyline/-gon
Record index number
Prepare multipart polygons
Augment idx for later on, & this trick eliminates trailing NaN rows
Record length
Shape type
Bounding box
Number of parts, number of points, part pointers
Write file length into .shp header
Close files
Check for dbfwrite function
Write rest of attributes
Attributes + shp data in mapstruct
Attributes + shp data in geostruct
```

7.8 shapewrite_man

7.9 shp2geo

7.10 shp2kml

7.11 shp_plot_attribute

7.12 split_section

7.13 write_polygon