dtwhaclustering

Release 1.0

Utpal Kumar

Aug 20, 2021

USAGE

CHAPTER

ONE

INSTALLATION

1.1 Requirements

1. `dtaidistance`: For computing DTW distance
2. `pygmt`: For plotting high-resolution maps
3. `pandas`: Analyze tabular data
4. `numpy`: Computation
5. `matplotlib`: Plotting time series
6. `scipy`: Interpolating data
7. `xarray`: Multilayered data structure

1.2 Install from PyPI

This package is available on PyPI (requires Python 3):

```
pip install dtwhaclustering
```

CHAPTER

TWO

DYNAMIC TIME WARPING BASED HIERARCHICAL AGGLOMERATIVE CLUSTERING

Codes to perform Dynamic Time Warping Based Hierarchical Agglomerative Clustering of GPS data

author Utpal Kumar

date 2021/08

CHAPTER

THREE

DTWHACLUSTERING.ANALYSIS_SUPPORT

DTW HAC analysis support functions (*analysis_support*)

author Utpal Kumar, Institute of Earth Sciences, Academia Sinica

dtwhaclustering.analysis_support.**dec2dt**(*start*)
Convert the decimal type time array to the date-time type array

Parameters **start** (*list*) – list or numpy array of decimal year values e.g., [2020.001]

Returns date-time type array

Return type list

dtwhaclustering.analysis_support.**dec2dt_scalar**(*st*)
Convert the decimal type time value to the date-time type

Parameters **st** – scalar decimal year value e.g., 2020.001

Returns time as datetime type

Return type str

dtwhaclustering.analysis_support.**toYearFraction**(*date*)
Convert the date-time type object to decimal year

Parameters **date** – the date-time type object

Returns decimal year

Return type float

CHAPTER

FOUR

DTWHACLUSTERING.PLOT_LINEAR_TREND

Plot linear trend values on a geographical map

author Utpal Kumar, Institute of Earth Sciences, Academia Sinica

dtwhaclustering.plot_linear_trend.**compute_interpolation**(*df*, *method='nearest'*, *lonrange=(120.0, 122.0)*, *latrange=(21.8, 25.6)*, *step=0.01*)

Interpolate linear trend values using Scipy's griddata and returns xarray object

Parameters

- **df** (*pandas.DataFrame*) – pandas dataframe containing the columns *lon*, *lat* and *slope*
- **method** (*str*) – Method of interpolation. One of {'linear', 'nearest', 'cubic'}. For more, see *scipy.interpolate.griddata*
- **lonrange** (*tuple*) – minimum and maximum values of the longitude to be interpolated
- **latrange** (*tuple*) – minimum and maximum values of the latitude to be interpolated
- **step** (*float*) – stepsize to interpolate data spatially

Returns *xarray.DataArray* of dims, and coords ("lat", "long"), maximum of the absolute interpolated array values

dtwhaclustering.plot_linear_trend.**plot_linear_trend_on_map**(*df*, *maplonrange=(120.0, 122.1)*, *maplatrange=(21.8, 25.6)*, *intrp_lonrange=(120.0, 122.0)*, *intrp_latrange=(21.8, 25.6)*, *outfig='Maps/slope-plot.png'*, *frame=['a1f0.25', 'WSen']*, *cmap='jet'*, *step=0.01*, *stn_labels=None*, *justify='left'*, *labelfont='6p,Helvetica-Bold,black'*, *offset='5p/-5p'*, *markerstyle='i10p'*, *defpen='1p,black'*, *stn_labels_color='black'*, *rand_justify=False*, *water_color='skyblue'*)

Plot the interpolated linear trend values along with the original data points on a geographical map using PyGMT

Parameters

- **df** (*pandas.DataFrame*) – Pandas dataframe containing the columns *lon*, *lat* and *slope*
- **maplonrange** (*tuple*) – longitude min/max of the output map
- **maplatrange** (*tuple*) – latitude min/max of the output map

- **intrp_lonrange** (*tuple*) – longitude min/max for the interpolation of data
- **intrp_latrange** (*tuple*) – latitude min/max for the interpolation of data
- **step** (*float*) – resolution of the interpolation
- **cmap** (*str*) – colormap for the output map
- **frame** (*list*) – frame of the output map. See PyGMT docs for details
- **outfig** (*str*) – output figure name with extension, e.g., *slope-plot.png*
- **water_color** (*str*) – color of the water, default: skyblue
- **stn_labels** (*array*) – label the station names. Only the stations provided will be labeled
- **justify** (*str*) – justification of the station labels - 'left', 'right'
- **rand_justify** (*boolean*) – randomly decide the justification for each labels
- **markerstyle** (*str*) – marker style and size

Returns None

CHAPTER

FIVE

DTWHACLUSTERING.LEASTSQUAREMODELING

Least square modeling of GPS displacements for seasonality, tidal, co-seismic jumps.

author Utpal Kumar, Institute of Earth Sciences, Academia Sinica

class dtwhaclustering.leastSquareModeling.**LSQmodules**(*dUU*, *sel_eq_file=None*, *station_loc_file='helper_files/stn_loc.txt'*, *comp='U'*, *figdir='LSQOut'*, *periods=(13.6608, 14.7653, 27.5546, 182.62, 365.26, 6793.836)*)

Bases: object

compute_lsq(*plot_results=False*, *remove_trend=True*, *remove_seasonality=True*, *remove_jumps=True*, *plotformat=None*)

Compute the least-squares model using multithreading

Parameters

- **plot_results** (*boolean*) – plot the final results
- **remove_trend** (*boolean*) – return the time series after removing the linear trend
- **remove_seasonality** (*boolean*) – return the time series after removing the seasonal signals
- **remove_jumps** (*boolean*) – return the time series after removing the co-seismic jumps
- **plotformat** (*str*) – plot format of the output figure, e.g. "png". "pdf" by default.

jump(*t*, *t0*)

heaviside step function

Parameters

- **t** (*list*) – time data
- **t0** – earthquake origin time

dtwhaclustering.leastSquareModeling.**lsqmodeling**(*dUU*, *dNN*, *dEE*, *stnlocfile*, *plot_results=True*, *remove_trend=False*, *remove_seasonality=True*, *remove_jumps=False*, *sel_eq_file='helper_files/selected_eqs_new.txt'*, *figdir='LSQOut'*)

Least square modeling for the three component time series

Parameters

- **dUU** (*pandas.DataFrame*) – Vertical component pandas dataframe time series
- **dNN** (*pandas.DataFrame*) – North component pandas dataframe time series
- **dEE** (*pandas.DataFrame*) – East component pandas dataframe time series

- **plot_results** (*boolean*) – plot the final results
- **remove_trend** (*boolean*) – return the time series after removing the linear trend
- **remove_seasonality** (*boolean*) – return the time series after removing the seasonal signals
- **remove_jumps** (*boolean*) – return the time series after removing the co-seismic jumps
- **sel_eq_file** (*str*) – File containing the earthquake origin times e.g., *2009,1,15* with header info, e.g. *year_val,month_val,date_val*
- **stnlocfile** (*str*) – File containing the station location info, e.g., *DAWU,120.89004,22.34059*, with header *stn,lon,lat*

Returns Pandas dataframe corresponding to the vertical, north and east components e.g., final_dU, final_dN, final_dE

Return type pandas.DataFrame, pandas.DataFrame, pandas.DataFrame

```
from dtwhaclustering.leastSquareModeling import lsqmodeling
final_dU, final_dN, final_dE = lsqmodeling(dUU, dNN, dEE,stnlocfile="helper_files/
↪stn_loc.txt",  plot_results=True, remove_trend=False, remove_seasonality=True,␣
↪remove_jumps=False)
```

CHAPTER

SIX

DTWHACLUSTERING.DTW_ANALYSIS

Classes and functions for the DTW analysis and plotting maps and figures (*dtw_analysis*) This module is built around the dtaidistance package for the DTW computation and scipy.cluster

author Utpal Kumar, Institute of Earth Sciences, Academia Sinica

note See dtaidistance for details on HierarchicalTree, dtw, dtw_visualisation

class dtwhaclustering.dtw_analysis.**dtw_clustering**(*matrix*, *labels=[]*, *longitudes=[]*, *latitudes=[]*)

Bases: object

compute_cluster(*clusterMatrix=None*, *window='constrained'*, *windowfrac=0.25*)

Parameters **compute_cluster** – data matrix to cluster

Returns model, cluster_idx

compute_cut_off_inconsistency(*t=None*, *depth=2*, *criterion='inconsistent'*, *return_cluster=False*)

Calculate inconsistency statistics on a linkage matrix following *scipy.cluster.hierarchy.inconsistent*. It compares each cluster merge's height *h* to the average *avg* and normalize it by the standard deviation *std* formed over the depth previous levels

Parameters

- **t** (*scalar*) – threshold to apply when forming flat clusters. See scipy.cluster.hierarchy.fcluster for details
- **depth** (*int*) – The maximum depth to perform the inconsistency calculation

Returns maximum inconsistency coefficient for each non-singleton cluster and its children; the inconsistency matrix (matrix with rows of avg, std, count, inconsistency); cluster

compute_dendrogram(*color_thresh=None*)

compute_distance_accl(*clusterMatrix=None*)

compute_distance_matrix(*compact=True*, *block=None*)

get_linkage(*clusterMatrix=None*)

optimum_cluster_elbow(*minmax=False*, *plotloc=False*)

Gives the optimum number of clusters required to express the maximum difference in the similarity using the elbow method

plot_cluster_geomap(*dtw_distance='optimal'*, *minlon=None*, *maxlon=None*, *minlat=None*, *maxlat=None*, *figname='dtw_cluster.pdf'*, *colorbar=True*, *colorbarstep=1*, *doffset=1*, *dpi=720*, *topo_data='@earth_relief_15s'*, *plot_topo=False*, *markerstyle='c0.3c'*, *cmap_topo='topo'*, *cmap_data='jet'*, *projection='M4i'*, *topo_cpt_range='-8000/8000/1000'*, *landcolor='#666666'*)

Plot the cluster points on a geographical map

Parameters

- **figname** (*str*) – output figure name
- **colorbar** (*bool*) – plot colorbar
- **colorbarstep** (*int*) – step for the colorbar
- **cmap** (*str*) – colormap for the cluster points
- **projection** (*str*) – map projection
- **topo_data** (*str*) – topographic data resolution, see pygmt docs for details
- **topo_cpt_range** – min/max/step for topographic color
- **landcolor** (*str*) – color for land region
- **dpi** (*int*) – output figure resolution

plot_cluster_geomap_interpolated(*dtw_distance='optimal'*, *lonrange=(120.0, 122.0)*, *latrange=(21.8, 25.6)*, *gridstep=0.01*, *figname='dtw_cluster_interp.pdf'*, *minlon=None*, *maxlon=None*, *minlat=None*, *maxlat=None*, *markerstyle='c0.3c'*, *dpi=720*, *doffset=1*, *plot_data=True*, *plot_intrp=True*)

Parameters

- **dtw_distance** (*str or float*) – use *dtw_distance* value to obtain the cluster division. If *optimal* then the optimal *dtw_distance* will be calculated
- **lonrange** (*tuple*) – minimum and maximum of longitude values for interpolation
- **latrange** (*tuple*) – minimum and maximum of latitude values for interpolation
- **gridstep** (*float*) – step size for interpolation

plot_cluster_xymap(*dtw_distance='optimal'*, *figname=None*, *xlabel='x'*, *ylabel='y'*, *colorbar=True*, *colorbarstep=1*, *scale=2*, *fontsize=26*, *markersize=12*, *axesfontsize=20*, *xtickstep=1*, *tickfontsize=20*, *edgecolors='k'*, *cmap='jet'*, *linewidths=2*, *cbarsize=18*)

Plot the cluster points in a rectangular coordinate system

Parameters

- **dtw_distance** (*str or float*) – use *dtw_distance* value to obtain the cluster division. If *optimal* then the optimal *dtw_distance* will be calculated
- **figname** (*str*) – output figure name
- **colorbar** (*bool*) – plot colorbar
- **colorbarstep** (*int*) – step for the colorbar
- **scale** (*int*) – figure scale
- **markersize** – cluster points size
- **edgecolors** – marker edge color

- **cmap** (*str*) – colormap for the cluster points

Returns figure, axes

plot_dendrogram(*figname=None*, *figsize=(20, 8)*, *xtickfontsize=26*, *labelfontsize=26*, *xlabel='3-D Stations'*, *ylabel='DTW Distance'*, *truncate_p=None*, *distance_threshold=None*, *annotate_above=0*, *plotpdf=True*, *leaf_rotation=0*)

Parameters **truncate_p** – show only last truncate_p out of all merged branches

plot_hac_iteration(*figname=None*, *figsize=(10, 8)*, *xtickfontsize=26*, *labelfontsize=26*, *xlabel='Iteration #'*, *ylabel='DTW Distance'*, *plot_color='C0'*, *plotpdf=True*)

plot_optimum_cluster(*max_d=None*, *figname=None*, *figsize=(10, 6)*, *plotpdf=True*, *xlabel='Number of Clusters'*, *ylabel='Distance'*, *accl_label='Curvature'*, *accl_color='C10'*, *dist_label='DTW Distance'*, *dist_color='k'*, *xlim=None*, *legend_outside=True*, *fontsize=26*, *xlabelfont=26*, *ylabelfont=26*)

Parameters **xlim** (*list*) – x limits of the plot e.g., [1,2]

plot_polar_dendrogram(*figsize=(20, 20)*, *normfactor=None*, *Nyticks=7*, *gap=0.05*, *Nsmooth=100*, *linewidth=1*, *xtickfontsize=20*, *ytickfontsize=20*, *plotstyle='seaborn'*, *figname=None*, *plotpdf=True*, *gridcolor=None*, *gridstyle='--'*, *gridwidth=1*, *tickfontweight='bold'*, *distance_threshold=None*)

Plot polar dendrogram of the clustering result

Parameters

- **figsize** – figure size
- **normfactor** – (optional) normalization factor for the log spacing between the yticks, -np.log(dcoord+normfactor)
- **Nyticks** – number of y ticks
- **gap** – gap for the yticks
- **plotstyle** – matplotlib plot style, *plotstyle* by default
- **distance_threshold** – str, float. threshold for the coloring of branches. If str=”optimal”, then the optimal number of clusters based on elbow method will be used

plot_signals(*figname=None*, *figsize=(10, 6)*, *fontsize=26*)

Parameters

- **figname** (*str*) – output figure name
- **figsize** (*tuple*) – output figure size

significance_test(*numsimulations=10*, *outfile='pickleFiles/dU_accl_sim_results.pickle'*, *fresh_start=False*)

class dtwhaclustering.dtw_analysis.**dtw_signal_pairs**(*s1*, *s2*, *labels=['s1', 's2']*)

Bases: object

compute_distance(*pruning=True*, *best_path=False*)

Returns the DTW distance

Parameters **pruning** (*boolean*) – prunes computations by setting max_dist to the Euclidean upper bound

compute_warping_path(*windowfrac=None*, *psi=None*, *fullmatrix=False*)
Returns the DTW path :param windowfrac: Fraction of the signal length. Only allow for shifts up to this amount away from the two diagonals. :param psi: Up to psi number of start and end points of a sequence can be ignored if this would lead to a lower distance :param full_matrix: The full matrix of all warping paths (or accumulated cost matrix) is built

plot_matrix(*windowfrac=0.2*, *psi=None*, *figname=None*, *shownumbers=False*, *showlegend=True*)
Plot the signals with the DTW matrix

Parameters **figname** (*str*) – output figure name

plot_signals(*figname=None*, *figsize=(12, 6)*)
Plot the signals

Parameters

- **figname** (*str*) – output figure name
- **figsize** (*tuple*) – output figure size

Returns figure and axes object

plot_warping_path(*figname=None*, *figsize=(12, 6)*)
Plot the signals with the warping paths

Parameters

- **figname** (*str*) – output figure name
- **figsize** (*tuple*) – output figure size

dtwhaclustering.dtw_analysis.**noise_robustness_test**(*clean_df*, *scale=0.1*)

dtwhaclustering.dtw_analysis.**plot_cluster**(*lons*, *lats*, *figname=None*, *figsize=(10, 10)*, *plotpdf=True*, *labels=[]*, *markersize=20*)

Parameters

- **figname** (*str*) – output figure name
- **figsize** (*tuple*) – output figure size

dtwhaclustering.dtw_analysis.**plot_signals**(*matrix*, *labels=[]*, *figname=None*, *plotpdf=True*, *figsize=(10, 8)*, *ylabelsize=26*, *color=None*)
Plot signals

Parameters

- **color** – list of colors. If None then matplotlib defaults color sequence will be used
- **figname** (*str*) – output figure name
- **figsize** (*tuple*) – output figure size

dtwhaclustering.dtw_analysis.**shuffle_signals**(*matrix*, *labels=[]*, *plot_signals=False*, *figsize=(10, 12)*, *figname=None*, *plotpdf=True*)

Parameters

- **figname** (*str*) – output figure name
- **figsize** (*tuple*) – output figure size

CHAPTER

SEVEN

DTWHACLUSTERING.PLOT_STATIONS

Plot topographic station map

author Utpal Kumar, Institute of Earth Sciences, Academia Sinica

dtwhaclustering.plot_stations.**plot_station_map**(*station_data*, *minlon=None*, *maxlon=None*, *minlat=None*, *maxlat=None*, *outfig='station_map.png'*, *datacolor='blue'*, *topo_data='@earth_relief_15s'*, *cmap='etopo1'*, *projection='M4i'*, *datalabel='Stations'*, *markerstyle='i10p'*, *random_station_label=False*, *stn_labels=None*, *justify='left'*, *labelfont='6p,Helvetica-Bold,black'*, *offset='5p/-5p'*, *stn_labels_color='red'*, *rand_justify=False*)

Plot topographic station map using PyGMT

Parameters

- **station_data** – Pandas dataframe containing columns *lon*, *lat*
- **minlon** – Minimum longitude of the map (optional)
- **maxlon** – Maximum longitude of the map (optional)
- **minlat** – Minimum latitude of the map (optional)
- **maxlat** – Maximum latitude of the map (optional)
- **datacolor** – color of the data point to plot
- **topo_data** – etopo data file
- **cmap** – colormap for the output topography
- **projection** – projection of the map. Mercator of width 4 inch by default
- **datalabel** – Label for the data
- **markerstyle** – Style of the marker. Inverted triangle of size 10p by default.
- **outfig** – Output figure path
- **random_station_label** – int. label randomly selected *random_station_label* of stations

```
from dtwhaclustering import plot_stations
plot_stations.plot_station_map(station_data = 'helper_files/selected_stations_info.
↪txt', outfig=f'{outloc}/station_map.pdf')
```

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

d

INDEX

P

S

T