MagPySV Documentation

Release 2.0

Grace Cox

May 04, 2021

Contents:

CHAPTER 1

Introduction

MagPySV is an open-source Python package designed to provide a consistent, and automated as far as possible, means of generating high resolution SV time series from raw observatory hourly means distributed by the World Data Centre (WDC) for Geomagnetism at the British Geological Survey (BGS), Edinburgh. It uses a second Python package developed by BGS to download data for a given time range and list of observatories.

The package may be run on the command line or within an interactive Jupyter notebook, and allows the user to obtain data in WDC format from BGS servers for a user-specified time period and list of observatories. It produces time series of the X, Y and Z components of the internally-generated magnetic field and secular variation (SV) at the desired frequency (typically monthly or annual means), and applies corrections for all documented baseline changes. Optionally, the user may exclude data using the ap index, which removes effects from documented high solar activity periods such as geomagnetic storms. Robust statistics are used to identify and remove outliers.

The software develops previously published denoising methods, which aim to remove external field contamination from the internal field, using principal component analysis, a method that uses the covariance matrix of the residual between the observed SV and that predicted by a global field model to create and remove a proxy for external field signal from the data. This method, based on Wardinski & Holme (2011), creates a single covariance matrix for all observatories of interest combined and applies the external field correction to all locations simultaneously, resulting in cleaner time series of the internally-generated SV.

Added in August 2020: a new module called GVO_tools, which was used to denoise Geomagnetic Virtual Observatories (GVOs) made from Swarm satellite data. This work was performed under a European Space Agency (ESA) Swarm DISC project on GVOs. More information on this project can be found at https://www.space.dtu.dk/english/research/projects/project-descriptions/geomagnetic-virtual-observatories

CHAPTER 2

Installation

MagPySV can be installed via the Python Package Index (PyPI) using the command `pip install magpysv`. This also installs all required dependencies, including the BGS data downloading app. The example Juypyter notebooks (see below) are not included in the PyPI installation and need to be downloaded from a separate GitHub project page after the software is installed.

CHAPTER 3

Example workflows

In the paper accompanying this software (in prep), we present two case studies of cleaned data in different geographic regions and discuss their application to geomagnetic jerks: monthly first differences for Europe, and annual differences for northern high latitude regions. A second GitHub repository accompanying this package includes two notebooks that can be used to download the relevant hourly data from BGS and reproduce the figures for these case studies. Note that MagPySV must be installed before using the notebooks (preferably using `pip`).

To obtain the notebooks, go to their GitHub project page at https://github.com/gracecox/MagPySV-examples and click the green "clone or download" button. From the dropdown menu, select "download zip" and save the file to the desired location. Unzip the file and navigate to the resulting folder. To open the notebooks, launch a Jupyter notebook from this folder (e.g. by typing `jupyter notebook` into the command line from the directory) and then select the desired notebook from the list that appears in your web browser.

The Jupyter documentation contains a step-by-step tutorial on installing and running notebooks, and is aimed at new users who have no familiarity with Python.

CHAPTER 4

Contributing

We hope others in the geomagnetism community find this code useful and welcome suggestions, feedback and contributions. Requesting new features or reporting bugs can be done by creating a GitHub issue for the repository.

If you would like to fix bugs or implement new features yourself, this is very welcome! This is done by

1. Forking MagPySV's GitHub repository
2. Creating a branch for your changes
3. Making your changes to the code
4. Submitting a pull request to the repository

Those unfamiliar with this process might find GitHub's tutorials useful. If this still looks too complicated, you are welcome to create a GitHub issue or get in touch with us directly for help.

CHAPTER 5

Reference

An open access manuscript describing MagPySV is now published in *Geochemistry, Geophysics, Geosystems* (https://doi.org/10.1029/2018GC007714); please cite this if you use the code. The paper also presents two case studies of cleaned data at European and high latitude observatories, and their application to geomagnetic jerks.

The project's main directory contains the readme file, license and various setup files. The *magpysv* directory contains all of the Python modules that make up the Python package, detailed descriptions of all functions contained within each module are found at the links below the contents section of this page. The code documentation can also be accessed using Python's help function by typing a command in this format: `help(magpysv.modulename.functionname)`

5.1 denoise module

Module containing functions to remove external signal from geomagnetic data.

Part of the MagPySV package for geomagnetic data analysis. This module provides various functions to denoise geomagnetic data by performing principal component analysis and identifying and removing outliers. Also contains an outlier detection function based on median absolute deviation from the median (MAD).

`magpysv.denoise.`**`detect_outliers`**(*, *dates*, *signal*, *obs_name*, *window_length*, *threshold*, *signal_type='SV'*, *plot_fig=False*, *save_fig=False*, *write_path=None*, *fig_size=(8, 6)*, *font_size=12*, *label_size=16*)

Detect outliers in a time series and remove them.

Use the median absolute deviation from the median (MAD) to identify outliers. The time series are long and highly variable so it is not appropriate to use single values of median to represent the whole series. The function uses a running median to better characterise the series (the window length and a threshold value stating many MADs from the median a point must be before it is classed as an outlier are user-specified).

Parameters

- **`dates`** (*datetime.datetime*) – dates of the time series measurements.

- **signal** (*array*) – array (or column from a pandas.DataFrame) containing the time series of interest.
- **obs_name** (*str*) – states the component of interest and the three digit IAGA observatory name.
- **window_length** (*int*) – number of months over which to take the running median.
- **threshold** (*float*) – the minimum number of median absolute deviations a point must be away from the median in order to be considered an outlier.
- **signal_type** (*str*) – specify whether magnetic field ('MF') or secular variation ('SV') is plotted. Defaults to SV.
- **plot_fig** (*bool*) – option to plot figure of the time series and identified outliers. Defaults to False.
- **save_fig** (*bool*) – option to save figure if plotted. Defaults to False.
- **write_path** (*str*) – output path for figure if saved.
- **fig_size** (*array*) – figure size in inches. Defaults to 8 inches by 6 inches.
- **font_size** (*int*) – font size for axes. Defaults to 12 pt.
- **label_size** (*int*) – font size for axis labels. Defaults to 16 pt.

Returns the input signal with identified outliers removed (set to NaN).

Return type signal (array)

magpysv.denoise.**eigenvalue_analysis**(*, *dates*, *obs_data*, *model_data*, *residuals*, *proxy_number=1*)

Remove external signal from SV data using principal Component Analysis.

Perform principal component analysis (PCA) on secular variation residuals (the difference between the observed SV and that predicted by a geomagnetic field model) calculated from annual differences of monthly means at several observatories. Uses masked arrays to discount missing data points and calculates the eigenvalues/vectors of the (3nx3n) covariance matrix for n observatories. The residuals are rotated into the eigendirections and denoised using the method detailed in Wardinski & Holme (2011, GJI, https://doi.org/10.1111/j.1365-246X.2011.04988.x). The SV residuals of the noisy component for all observatories combined are used as a proxy for the unmodelled external signal. The denoised data are then rotated back into geographic coordinates. The PCA algorithm outputs the eigenvalues sorted from largest to smallest, so the corresponding eigenvector matrix has the 'noisy' direction in the first column and the 'clean' direction in the final column.

This algorithm masks missing data so that they are not taken into account during the PCA. Missing values are not infilled or estimated, so NaN values in the input dataframe are given as NaN values in the output.

Smallest eigenvalue 'quiet' direction

Largest eiegenvalue 'noisy' direction

Parameters

- **dates** (*datetime.datetime*) – dates of the time series measurements.
- **obs_data** (*pandas.DataFrame*) – dataframe containing columns for monthly/annual means of the X, Y and Z components of the secular variation at the observatories of interest.
- **model_data** (*pandas.DataFrame*) – dataframe containing columns for field model prediction of the X, Y and Z components of the secular variation at the same observatories as in obs_data.
- **residuals** (*pandas.DataFrame*) – dataframe containing the SV residuals (difference between the observed data and model prediction).

- **proxy_number** (*int*) – the number of 'noisy' directions used to create the proxy for the external signal removal. Default value is 1 (only the residual in the direction of the largest eigenvalue is used). Using n directions means that proxy is the sum of the SV residuals in the n noisiest eigendirections.

Returns

tuple containing:

- **denoised_sv (*pandas.DataFrame*):** dataframe with datetime objects in the first column and columns for the denoised X, Y and Z SV components at each of the observatories for which data were provided.
- **proxy (*array*):** the signal that was used as a proxy for unmodelled external magnetic field in the denoising stage.
- **eig_values (*array*):** the eigenvalues of the obs_data matrix.
- **eig_vectors (*array*):** the eigenvectors associated with the n largest eigenvalues of the data matrix. For example, if the residuals in the two 'noisiest' directions are used as the proxy for external signal, then these two eigenvectors are returned.
- **projected_residuals (*array*):** SV residuals rotated into the eigendirections.
- **corrected_residuals (*array*):** SV residuals after the denoising process.
- covariance_matrix (*array*): residuals covariance matrix.

Return type (tuple)

magpysv.denoise.**eigenvalue_analysis_impute**(*, *dates*, *obs_data*, *model_data*, *residuals*, *proxy_number=1*)

Remove external signal from SV data using Principal Component Analysis.

Perform principal component analysis (PCA) on secular variation residuals (the difference between the observed SV and that predicted by a geomagnetic field model) calculated from annual differences of monthly means at several observatories. Uses the imputer from sklearn.preprocessing to fill in missing data points and calculates the singular values of the data matrix for n observatories (uses Singular Values Decomposition, SVD). The residuals are rotated into the eigendirections and denoised using the method detailed in Wardinski & Holme (2011, GJI, https://doi.org/10.1111/j.1365-246X.2011.04988.x). The SV residuals of the noisy component for all observatories combined are used as a proxy for the unmodelled external signal. The denoised data are then rotated back into geographic coordinates. The pca algorithm outputs the singular values (these are equal to the square root of the eigenvalues of the covariance matrix) sorted from largest to smallest, so the corresponding eigenvector matrix has the 'noisy' direction in the first column and the 'clean' direction in the final column.

Note that the SVD algorithm cannot be used if any data are missing, which is why imputation is needed with this method. The function denoise.eigenvalue_analysis permits missing values and does not infill them - that is the more robust function.

Smallest eigenvalue: 'quiet' direction

Largest eiegenvalue: 'noisy' direction

Parameters

- **dates** (*datetime.datetime*) – dates of the time series measurements.
- **obs_data** (*pandas.DataFrame*) – dataframe containing columns for monthly/annual means of the X, Y and Z components of the secular variation at the observatories of interest.
- **model_data** (*pandas.DataFrame*) – dataframe containing columns for field model prediction of the X, Y and Z components of the secular variation at the same observatories as in obs_data.

- **residuals** (*pandas.DataFrame*) – dataframe containing the SV residuals (difference between the observed data and model prediction).
- **proxy_number** (*int*) – the number of 'noisy' directions used to create the proxy for the external signal removal. Default value is 1 (only the residual in the direction of the largest eigenvalue is used). Using n directions means that proxy is the sum of the SV residuals in the n noisiest eigendirections.

Returns

tuple containing:

- **denoised_sv (*pandas.DataFrame*):** dataframe with dates in the first column and columns for the denoised X, Y and Z secular variation components at each of the observatories for which data were provided.
- **proxy (*array*):** the signal that was used as a proxy for unmodelled external magnetic field in the denoising stage.
- **eig_values (*array*):** the singular values of the obs_data matrix.
- **eig_vectors (*array*):** the eigenvectors associated with the n largest singular values of the data matrix. For example, if the residuals in the two 'noisiest' directions are used as the proxy for external signal, then these two eigenvectors are returned.
- **projected_residuals (*array*):** SV residuals rotated into the eigendirections.
- **corrected_residuals (*array*):** SV residuals after the denoising process.

Return type (tuple)

5.2 gvo_tools module

5.3 io module

Module containing functions to parse World Data Centre (WDC) files.

Part of the MagPySV package for geomagnetic data analysis. This module provides various functions to read, parse and manipulate the contents of World Data Centre (WDC) formatted files containing geomagnetic data and output data to comma separated values (CSV) files. Also contains functions to read output of code used for the COV-OBS magnetic field model series by Gillet et al. (links below).

magpysv.io.**ae_parsefile**(*fname*)

Load a WDC-like format AE file and place contents into a dataframe.

Load a file of AE (Auroral Electroject) index hourly data in the format distributed by the Kyoto WDC at http://wdc.kugi.kyoto-u.ac.jp/dstae/index.html and extract the contents.

Parameters **fname** (*str*) – path to a WDC-like formatted AE file.

Returns dataframe containing hourly AE data. First column is a series of datetime objects (in the format yyyy-mm-dd hh:30:00) and second column contains theAE values at the specified times.

Return type data (pandas.DataFrame)

magpysv.io.**ae_readfile**(*fname*)

Wrapper function to call ae_parsefile and wdc_datetimes.

Parameters **fname** (*str*) – path to a AE file in Kyoto WDC-like format. Assumes data for all years are contained within this file.

Returns dataframe containing the data read from the WDC file. First column is a series of datetime objects (in the format yyyy-mm-dd hh:30:00) and second column contains AE values at the specified times (hourly means).

Return type data (pandas.DataFrame)

magpysv.io.**angles_to_geographic**(*data*)

Use D and H values to calculate the X and Y field components.

The declination (D) and horizontal intensity (H) relate to the north (Y) and east (X) components as follows:

X = H*cos(D)

Y = H*sin(D)

Parameters **data** (*pandas.DataFrame*) – dataframe containing columns for datetime objects and hourly means of the magnetic field components (D, I, F, H, X, Y or Z).

Returns the same dataframe with datetime objects in the first column and hourly means of the field components in either nT or degrees (depending on the component).

Return type data (pandas.DataFrame)

magpysv.io.**ap_readfile**(*fname*)

Load an kp/ap file and place the hourly ap values into a dataframe.

Load a datafile of 3-hourly ap data and extract the contents. Each of the 3-hourly values for a given day is repeated three times to give an hourly mean for all 24 hours of the day. This function is designed to read files downloaded from the GFZ, Potsdam server at ftp://ftp.gfz-potsdam.de/pub/home/obs/kp-ap/.

Parameters **fname** (*str*) – path to an ap datafile.

Returns dataframe containing hourly ap data. First column is a series of datetime objects (in the format yyyy-mm-dd hh:30:00) and second column contains ap values at the specified times.

Return type data (pandas.DataFrame)

magpysv.io.**append_ae_data**(*ae_data_path*)

Append AE data into a single dataframe containing all years.

Data downloaded from ftp://ftp.ngdc.noaa.gov/STP/GEOMAGNETIC_DATA/INDICES/AURORAL_ELECTROJET/HOURLY/ come in WDC-like format files with one file per year named aeyyyy.wdc (data provided by the WDC at Kyoto. Can be downloaded directly from http://wdc.kugi.kyoto-u.ac.jp/dstae/index.html)

Parameters **ae_data_path** (*str*) – path to directory containing WDC-like format AE datafiles. All AE files should be located in the same directory.

Returns dataframe containing all available hourly AE data. First column is a series of datetime objects (in the format yyyy-mm-dd hh:30:00) and second column contains AE values at the specified times.

Return type data (pandas.DataFrame)

magpysv.io.**append_ap_data**(*ap_data_path*)

Append ap data into a single dataframe containing all years.

Data downloaded from ftp://ftp.gfz-potsdam.de/pub/home/obs/kp-ap/wdc/ come in WDC-like format files with one file per year named kpyyyy.wdc. This function concatenates all years into a single dataframe.

Parameters **ap_data_path** (*str*) – path to directory containing WDC-like format ap datafiles. All ap files should be located in the same directory.

Returns dataframe containing all available hourly ap data. First column is a series of datetime objects (in the format yyyy-mm-dd hh:30:00) and second column contains ap values at the specified times.

Return type data (pandas.DataFrame)

magpysv.io.**append_wdc_data**(*, *obs_name*, *path=None*)

Append all WDC data for an observatory into a single dataframe.

Parameters

- **obs_name** (*str*) – observatory name (as 3-digit IAGA code).
- **path** (*str*) – path to directory containing WDC datafiles. All files for the observatory should be located in the same directory.

Returns dataframe containing all available hourly geomagnetic data at a single observatory. First column is a series of datetime objects (in the format yyyy-mm-dd hh:30:00) and subsequent columns are the X, Y and Z components of the magnetic field at the specified times.

Return type data (pandas.DataFrame)

magpysv.io.**combine_csv_data**(*, *start_date*, *end_date*, *sampling_rate='MS'*, *obs_list*, *data_path*, *model_path*, *day_of_month=1*)

Read and combine observatory and model SV data for several locations.

Calls read_csv_data to read observatory data and field model predictions for each observatory in a list. The data and predictions for individual observatories are combined into their respective large dataframes. The first column contains datetime objects and subsequent columns contain X, Y and Z secular variation/field components (in groups of three) for all observatories.

Parameters

- **start_date** (*datetime.datetime*) – the start date of the data analysis.
- **end_date** (*datetime.datetime*) – the end date of the analysis.
- **sampling_rate** (*str*) – the sampling rate for the period of interest. The default is 'MS', which creates a range of dates between the specified values at monthly intervals with the day fixed as the first of each month. Use 'M' for the final day of each month. Other useful options are 'AS' (a series of dates at annual intervals, with the day and month fixed at 01 and January respectively) and 'A' (as for 'AS' but with the day/month fixed as 31 December.)
- **obs_list** (*list*) – list of observatory names (as 3-digit IAGA codes).
- **data_path** (*str*) – path to the CSV files containing observatory data.
- **model_path** (*str*) – path to the CSV files containing model SV data.
- **day_of_month** (*int*) – For SV data, first differences of monthly means have dates at the start of the month (i.e. MF of mid-Feb minus MF of mid-Jan should give SV at Feb 1st. For annual differences of monthly means the MF of mid-Jan year 2 minus MF of mid-Jan year 1 gives SV at mid-July year 1. The dates of COV-OBS output default to the first day of the month (compatible with dates of monthly first differences SV data, but not with those of annual differences). This option is used to set the day part of the dates column if required. Default to 1 (all output dataframes will have dates set at the first day of the month.)

Returns

tuple containing:

- **obs_data (*pandas.DataFrame*):** dataframe containing SV data for all observatories in obs_list.
- **model_sv_data (*pandas.DataFrame*):** dataframe containing SV predictions for all observatories in obs_list.

- **model_mf_data (*pandas.DataFrame*):** dataframe containing magnetic field predictions for all observatories in obs_list.

Return type (tuple)

magpysv.io.**covobs_datetimes**(*data*)

Create datetime objects from COV-OBS field model output file.

The format output by the field model is year.decimalmonth e.g. 1960.08 is Jan 1960.

Parameters **data** (*pandas.DataFrame*) – needs a column for decimal year (in yyyy.mm format).

Returns the same dataframe with the decimal year column replced with a series of datetime objects in the format yyyy-mm-dd.

Return type data (pandas.DataFrame)

magpysv.io.**covobs_parsefile**(*, *fname*, *data_type*)

Loads MF and SV predictions from the COV-OBS geomagnetic field model.

Load a datafile containing SV/MF predictions from the COV-OBS magnetic field model series by Gillet et al. (2013, Geochem. Geophys. Geosyst., https://doi.org/10.1002/ggge.20041; 2015, Earth, Planets and Space, https://doi.org/10.1186/s40623-015-0225-z) field model.

Parameters

- **fname** (*str*) – path to a COV-OBS datafile.
- **data_type** (*str*) – specify whether the file contains magnetic field data ('mf') or or secular variation data ('sv')

Returns dataframe containing hourly geomagnetic data. First column is a series of datetime objects (in the format yyyy-mm-dd) and subsequent columns are the X, Y and Z components of the SV/MF at the specified times.

Return type model_data (pandas.DataFrame)

magpysv.io.**covobs_readfile**(*, *fname*, *data_type*)

Wrapper function to call covobs_parsefile and covobs_datetimes.

The COV-OBS code (publically available) can be used to produce synthetic observatory time series for other field models if the appropriate spline file is used. The output will be of the same format as COV-OBS output and can be read using MagPySV.

Parameters

- **fname** (*str*) – path to a COV-OBS format datafile.
- **data_type** (*str*) – specify whether the file contains magnetic field data ('mf') or or secular variation data ('sv')

Returns dataframe containing the data read from the file. First column is a series of datetime objects (in the format yyyy-mm-dd) and subsequent columns are the X, Y and Z components of the SV/MF at the specified times.

Return type data (pandas.DataFrame)

magpysv.io.**datetime_to_decimal**(*date*)

Convert a datetime object to a decimal year.

Parameters **date** (*datetime.datetime*) – datetime object representing an observation time.

Returns the same date represented as a decimal year.

Return type date (float)

magpysv.io.**hourly_mean_conversion**(*data*)
Use the tabular base to calculate hourly means in nT or degrees (D, I).

Uses the tabular base and hourly value from the WDC file to calculate the hourly means of magnetic field components. Value is in nT for H, F, X, Y or Z components and in degrees for D or I components. Called by wdc_xyz.

hourly_mean = tabular_base*100 + wdc_hourly_value (for components in nT)

hourly_mean = tabular_base + wdc_hourly_value/600 (for D and I components)

Parameters **data** (*pandas.DataFrame*) – dataframe containing columns for datetime objects, magnetic field component (D, I, F, H, X, Y or Z), the tabular base and hourly mean.

Returns dataframe with datetime objects in the first column and hourly means of the field components in either nT or degrees (depending on the component).

Return type obs_data (pandas.DataFrame)

magpysv.io.**read_csv_data**(*, *fname*, *data_type*)
Read dataframe from a CSV file.

Parameters

- **fname** (*str*) – path to a CSV datafile.
- **data_type** (*str*) – specify whether the file contains magnetic field data ('mf') or or secular variation data ('sv')

Returns dataframe containing the data read from the CSV file.

Return type data (pandas.DataFrame)

magpysv.io.**separate_hourly_vals**(*hourstring*)
Separate individual hourly field means from the string containing all 24 values in the WDC file. Called by wdc_parsefile.

Parameters **hourstring** (*str*) – string containing the hourly magnetic field means parsed from a WDC file for a single day.

Returns list containing the hourly field values.

Return type hourly_vals_list (list)

magpysv.io.**separate_hourly_vals_ae**(*hourstring*)
Separate individual hourly field means from the string containing all 24 values in the AE file. Called by ae_parsefile.

Parameters **hourstring** (*str*) – string containing the hourly AE means parsed from a Kyoto WDC-like file for a single day.

Returns list containing the hourly AE values.

Return type hourly_vals_list (list)

magpysv.io.**separate_three_hourly_vals**(*hourstring*)
Separate 3-hourly ap means from the string containing all 8 values.

Separate the 8 individual 3-hourly ap means from the string containing all values for the day. Each value is repeated 3 times to give a value for each hour. Called by ap_readfile.

Parameters **hourstring** (*str*) – string containing the 3-hourly ap means parsed from a Kyoto WDC-like file for a single day.

Returns list containing the hourly ap values.

Return type hourly_vals_list (list)

magpysv.io.**wdc_datetimes**(*data*)
Create datetime objects from the fields extracted from a WDC datafile.

Parameters **data** (*pandas.DataFrame*) – needs columns for century, year (yy format), month, day and hour. Called by wdc_parsefile.

Returns the same dataframe with a series of datetime objects (in the format yyyy-mm-dd hh:30:00) in the first column.

Return type data (pandas.DataFrame)

magpysv.io.**wdc_parsefile**(*fname*)
Load a WDC datafile and place the contents into a dataframe.

Load a datafile of WDC hourly geomagnetic data for a single observatory and extract the contents. Parses the current WDC file format, but not the previous format containing international quiet (Q) or disturbed (D) day designation in place of the century field - only the newer format is downloaded from the BGS servers. Detailed file format description can be found at http://www.wdc.bgs.ac.uk/catalog/format.html

Parameters **fname** (*str*) – path to a WDC datafile.

Returns dataframe containing hourly geomagnetic data. First column is a series of datetime objects (in the format yyyy-mm-dd hh:30:00) and subsequent columns are the X, Y and Z components of the magnetic field at the specified times.

Return type data (pandas.DataFrame)

magpysv.io.**wdc_readfile**(*fname*)
Wrapper function to call wdc_parsefile, wdc_datetimes and wdc_xyz.

Parameters **fname** (*str*) – path to a WDC datafile.

Returns dataframe containing the data read from the WDC file. First column is a series of datetime objects (in the format yyyy-mm-dd hh:30:00) and subsequent columns are the X, Y and Z components of the magnetic field at the specified times (hourly means).

Return type data (pandas.DataFrame)

magpysv.io.**wdc_to_hourly_csv**(*, *wdc_path=None*, *write_dir*, *obs_list*, *print_obs=True*)
Convert WDC file to X, Y and Z hourly means and save to CSV file.

Finds WDC hourly data files for all observatories in a directory path (assumes data for all observatories are located inside the same directory). The BGS downloading app distributes data inside a single directory with the naming convention obsyear.wdc where obs is a three digit observatory name in lowercase and year is a four digit year, e.g. ngk1990.wdc or clf2013.wdc. This function converts the hourly WDC format data to hourly X, Y and Z means, appends all years of data for a single observatory into a single dataframe and saves the dataframe to a CSV file.

Parameters

- **wdc_path** (*str*) – path to the directory containing datafiles.
- **write_dir** (*str*) – directory path to which the output CSV files are written.
- **obs_list** (*list*) – list of observatory names (as 3-digit IAGA codes).
- **print_obs** (*bool*) – choose whether to print each observatory name as the function goes through the directories. Useful for checking progress as it can take a while to read the whole WDC dataset. Defaults to True.

magpysv.io.**wdc_xyz**(*data*)
Convert extracted WDC data to hourly X, Y and Z components in nT.

Missing values (indicated by 9999 in the datafiles) are replaced with NaNs.

Parameters **data** (*pandas.DataFrame*) – dataframe containing columns for datetime objects, magnetic field component (D, I, F, H, X, Y or Z), the tabular base and hourly mean.

Returns the same dataframe with datetime objects in the first column and columns for X, Y and Z components of magnetic field (in nT).

Return type data (pandas.DataFrame)

magpysv.io.**write_csv_data**(*, *data*, *write_dir*, *obs_name*, *file_prefix=None*, *decimal_dates=False*, *header=True*)

Write dataframe to a CSV file.

Parameters

- **data** (*pandas.DataFrame*) – data to be written to file.
- **write_dir** (*str*) – directory path to which the output CSV file is written.
- **obs_name** (*str*) – name of observatory at which the data were obtained.
- **file_prefix** (*str*) – optional string to prefix the output CSV filenames (useful for specifying parameters used to create the dataset etc).
- **decimal_dates** (*bool*) – optional argument to specify that dates should be written in decimal format rather than datetime objects. Defaults to False.
- **header** (*bool*) – option to include header in file. Defaults to True.

5.4 model_prediction module

Module containing functions to run the COV-OBS (Gillet et al) field model.

Part of the MagPySV package for geomagnetic data analysis. Contains a function to obtain a complete list of geomagnetic observatory locations from the WDC webserver and another function to run the COV-OBS magnetic field model by Gillet et al. (2013, Geochem. Geophys. Geosyst., https://doi.org/10.1002/ggge.20041; 2015, Earth, Planets and Space, https://doi.org/10.1186/s40623-015-0225-z2013) to obtain model predictions for these observatory locations. The code can be obtained from http://www.spacecenter.dk/files/magnetic-models/COV-OBSx1/ and no modifications are necessary to run it using MagPySV.

magpysv.model_prediction.**get_observatory_list**()

Obtain the complete list of observatory locations held by the WDC.

Obtains a dictionary containing information for all geomagnetic observatories known to the WDC using the BGS website at: http://app.geomag.bgs.ac.uk/wdc/

The following information is given for each location:

'AAA': { 'code': 'AAA',

'country': 'Kazakhstan',

'dataAvailability': {'hour': {'earliest': 1963, 'latest': 2015},

'minute': {'earliest': 2005, 'latest': 2015}},

'dateClosed': None,

'dateOpened': [1963, 1, 1],

'elevation': 1300.0,

'latitude': 43.18,

'longitude': 76.92,

'name': 'Alma Ata' }

Returns dictionary containing information about each geomagnetic observatory.

Return type stations (dict)

magpysv.model_prediction.**run_covobs**(*, *stations*, *model_path*, *output_path*)

Use observatory latitude, longitude and elevation to run COV-OBS.

Uses the dictionary of observatory information obtained from the WDC site to run the COV-OBS field model Gillet et al. (2013, Geochem. Geophys. Geosyst., https://doi.org/10.1002/ggge.20041; 2015, Earth, Planets and Space, https://doi.org/10.1186/s40623-015-0225-z2013) for a given location given in geodetic coordinates (model output is also in geodetic coordinates). Converts latitude in degrees to colatitude in radians, longitude in degrees (0 to 360) into radians (-pi to pi) and elevation in m to km. It then runs the fortran exectuable for the field model and passes the location data as command line arguments. The output files are stored as mf_obs.dat and sv_obs.dat for magnetic field and secular variation predictions respectively (e.g. mf_ngk.dat and sv_ngk.dat for Niemegk.)

Assumes that the user has compiled the fortran source code and called the executable "a.out", e.g. has run

$ gfortran cov-obs-coefs.f

No modification to the fortran source code is required (code can be downloaded from http://www.spacecenter.dk/files/magnetic-models/COV-OBSx1/). The file variables set as *parameter* in *cov-obs.x1-prog2CF/param.dat* should be altered to the time range and sampling rate desired, and importantly, *IFUNC=2* should be set to ouput time series values at a location.

The COV-OBS code can also be used to run other field models if modified to accept a different spline file as input, rather than the supplied COV-OBS.x1-int file.

Parameters

- **stations** (*dict*) – dictionary containing information about each geomagnetic observatory.
- **model_path** (*str*) – path to the compiled COV-OBS executable.
- **output_path** (*str*) – path to the directory in which the model output should be stored.

5.5 plots module

Module containing plotting functions.

Part of the MagPySV package for geomagnetic data analysis. This module provides various plotting functions.

magpysv.plots.**compare_proxies**(*, *fname1*, *fname2*, *legend_text*, *fig_size=(8, 6)*, *font_size=12*, *label_size=16*, *save_fig=False*, *write_path=None*)

Compare proxies of unmodelled external signal for different analyses.

Calculates the correlation coefficients of two given proxies for unmodelled external signals and includes it on a plot of the two series. Each proxy is formed of the SV residuals projected into the eigendirection(s) of the largest eigenvalues of the residual covariance matriloc. The proxies are reduced to zero-mean and unit-variance on the plots (zscore).

Parameters

- **fname1** (*str*) – path to file containing a time series of proxy for noise.
- **fname2** (*str*) – path to a second file containing a proxy for noise.

- **legend_text** (*str*) – text to include on the plot legend.
- **fig_size** (*array*) – figure size in inches. Defaults to 8 inches by 6 inches.
- **font_size** (*int*) – font size for axes. Defaults to 12 pt.
- **label_size** (*int*) – font size for axis labels. Defaults to 16 pt.
- **save_fig** (*bool*) – option to save figure. Defaults to False.
- **write_path** (*str*) – output path for figure if saved.

magpysv.plots.**plot_eigenvalues**(*, *values*, *fig_size=(8, 6)*, *font_size=12*, *label_size=16*, *save_fig=False*, *write_path=None*)

Plot eigenvalues of the covariance matrix of SV residuals.

Produces a plot of the eigenvalues obtained during the principal component analysis (PCA) of SV residuals. The largest eigenvalue represents the eigendirection with the largest contribution to the residuals (i.e. the "noisy" direction.). The smallest eigenvalue represents the eigendirection with the smallest contribution to the residuals (the "clean" direction). See Wardinski & Holme (2011, GJI, https://doi.org/10.1111/j.1365-246X.2011.04988.x) for further details.

Parameters

- **values** (*array*) – the eigenvalues obtained from the principal component analysis of the SV residuals.
- **fig_size** (*array*) – figure size in inches. Defaults to 8 inches by 6 inches.
- **font_size** (*int*) – font size for axes. Defaults to 12 pt.
- **label_size** (*int*) – font size for axis labels. Defaults to 16 pt.
- **save_fig** (*bool*) – option to save figure. Defaults to False.
- **write_path** (*str*) – output path for figure if saved.

magpysv.plots.**plot_eigenvectors**(*, *obs_names*, *eigenvecs*, *fig_size=(8, 6)*, *font_size=12*, *label_size=16*, *save_fig=False*, *write_path=None*)

Plot eigenvectors of the covariance matrix of SV residuals.

Produces a plot of the eigenvectors corresponding to the n largest eigenvalues of the covariance matrix obtained during PCA of SV residuals, where n is the number of eigenvalues used as a proxy for unmodelled external field signal. The n eigenvectors corresponding to the n largest eigenvalue represent the directions with the largest contribution to the residuals (i.e. the "noisiest" directions). See Wardinski & Holme (2011, GJI, https://doi.org/10.1111/j.1365-246X.2011.04988.x) for further details.

Parameters

- **obs_names** (*list*) – list of observatory names given as three digit IAGA codes.
- **eigenvecs** (*array*) – the eigenvalues obtained from the principal
- **analysis of the SV residuals.** (*component*) –
- **fig_size** (*array*) – figure size in inches. Defaults to 8 inches by 6 inches.
- **font_size** (*int*) – font size for axes. Defaults to 12 pt.
- **label_size** (*int*) – font size for axis labels. Defaults to 16 pt.
- **save_fig** (*bool*) – option to save figure. Defaults to False.
- **write_path** (*str*) – output path for figure if saved.

magpysv.plots.**plot_index**(*, *index_file*, *dates*, *projected_residuals*, *fig_size=(8, 6)*, *font_size=12*, *label_size=16*, *plot_legend=True*, *save_fig=False*, *write_path=None*, *index_name='Dst'*)

Compare the proxy used to denoise the SV data with a geomagnetic index.

Loads geomagnetic index and plots it alongside the signal used as a proxy for unmodelled external signal. Both time series are reduced to zero mean and unit variance (i.e. their zscore) for plotting.

Parameters

- **dates** (*datetime.datetime*) – dates of time series measurements.
- **index_file** (*str*) – path to the file containing index data.
- **projected_residuals** (*time series*) – difference between modelled and SV rotated into the eigendirections obtained during denoising (principal component analysis). The proxy for unmodelled external signal is the residual projected in the noisiest eigendirection(s).
- **index_name** (*str*) – name of index used in comparison e.g. Dst or ap. Defaults to Dst.
- **fig_size** (*array*) – figure size in inches. Defaults to 8 inches by 6 inches.
- **font_size** (*int*) – font size for axes. Defaults to 12 pt.
- **label_size** (*int*) – font size for axis labels. Defaults to 16 pt.
- **plot_legend** (*bool*) – option to include a legend on the plot. Defaults to True.
- **save_fig** (*bool*) – option to save figure. Defaults to False.
- **write_path** (*str*) – output path for figure if saved.

magpysv.plots.**plot_index_dft**(*, *index_file*, *dates*, *signal*, *fig_size=(8, 6)*, *font_size=12*, *label_size=16*, *plot_legend=True*, *save_fig=False*, *write_path=None*, *index_name='Dst'*)

Compare the DFTs of the proxy signal with that of a geomagnetic index.

Loads index data, calculates its DFT using an FFT algorithm and plots it alongside the DFT of the signal used as a proxy for unmodelled external signal. The length of the time series are padded with zeroes up to the next power of two.

Parameters

- **dates** (*datetime.datetime*) – dates of time series measurements.
- **signal** (*time series*) – proxy for unmodelled external signal used in the denoising process (principal component analysis). The proxy is the residual in the noisiest eigendirection(s).
- **index_file** (*str*) – path to the file containing index data.
- **index_name** (*str*) – name of index used in comparison e.g. Dst or Dcx. Defaults to Dst.
- **fig_size** (*array*) – figure size in inches. Defaults to 8 inches by 6 inches.
- **font_size** (*int*) – font size for axes. Defaults to 12 pt.
- **label_size** (*int*) – font size for axis labels. Defaults to 16 pt.
- **plot_legend** (*bool*) – option to include a legend on the plot. Defaults to True.
- **save_fig** (*bool*) – option to save figure. Defaults to False.
- **write_path** (*str*) – output path for figure if saved.

magpysv.plots.**plot_mf**(*, *dates*, *mf*, *model*, *obs*, *model_name*, *fig_size=(8, 6)*, *font_size=12*, *label_size=16*, *plot_legend=True*, *save_fig=False*, *write_path=None*)

Plot the SV and model prediction for a single observatory.

Produces a plot of the X, Y and Z components of the SV and field model prediction for a single observatory.

Parameters

- **dates** (*datetime.datetime*) – dates of time series measurements.
- **mf** (*time series*) – X, Y and Z components of magnetic field at a single location.
- **model** (*time series*) – X, Y and Z components of the field predicted by a field model for the same location as the data.
- **obs** (*str*) – observatory name given as three digit IAGA code.
- **model_name** (*str*) – field model name.
- **fig_size** (*array*) – figure size in inches. Defaults to 8 inches by 6 inches.
- **font_size** (*int*) – font size for axes. Defaults to 12 pt.
- **label_size** (*int*) – font size for axis labels. Defaults to 16 pt.
- **plot_legend** (*bool*) – option to include a legend on the plot. Defaults to True.
- **save_fig** (*bool*) – option to save figure. Defaults to False.
- **write_path** (*str*) – output path for figure if saved.

magpysv.plots.**plot_outliers**(*, *dates*, *signal*, *obs_name*, *outliers*, *signal_type='SV'*, *fig_size=(8, 6)*, *font_size=12*, *label_size=16*, *save_fig=False*, *write_path=None*)

Plot the SV and identified outliers.

Parameters

- **dates** (*datetime.datetime*) – dates of time series measurements.
- **signal** (*time series*) – single component of SV at a single location.
- **obs_name** (*str*) – states the SV component and observatory name given as three digit IAGA code. For example, the X component at NGK would be x_ngk if obs_name is taken from the pandas.DataFrame containing SV data for all observatories combined.
- **outliers** (*array*) – outliers identified by the denoise.detect_outliers function
- **signal_type** (*str*) – specify whether magnetic field ('MF') or secular variation ('SV') is plotted. Defaults to SV.
- **fig_size** (*array*) – figure size in inches. Defaults to 8 inches by 6 inches.
- **font_size** (*int*) – font size for axes. Defaults to 12 pt.
- **label_size** (*int*) – font size for axis labels. Defaults to 16 pt.
- **save_fig** (*bool*) – option to save figure. Defaults to False.
- **write_path** (*str*) – output path for figure if saved.

magpysv.plots.**plot_residuals_dft**(*, *projected_residuals*, *dates*, *fig_size=(10, 8)*, *font_size=12*, *label_size=16*, *plot_legend=True*, *save_fig=False*, *write_path=None*)

Compare the DFTs of the projected residuals with each other.

Calculates the DFT of the residuals in each eigendirection given and plots it alongside the residuals themselves. Produces a single figure with each eigendirection included as a subplot. Use plot_residuals_dft_all if a separate

figure per eigendirection is desired. The length of the time series are padded with zeroes up to the next power of two.

Parameters

- **dates** (*datetime.datetime*) – dates of time series measurements.
- **projected_residuals** (*time series*) – difference between modelled and SV rotated into the eigendirections obtained during denoising (principal component analysis). The proxy for unmodelled external signal is the residual projected in the noisiest eigendirection(s).
- **fig_size** (*array*) – figure size in inches. Defaults to 8 inches by 6 inches.
- **font_size** (*int*) – font size for axes. Defaults to 12 pt.
- **label_size** (*int*) – font size for axis labels. Defaults to 16 pt.
- **plot_legend** (*bool*) – option to include a legend on the plot. Defaults to True.
- **save_fig** (*bool*) – option to save figure. Defaults to False.
- **write_path** (*str*) – output path for figure if saved.

magpysv.plots.**plot_residuals_dft_all**(**, projected_residuals, dates, fig_size=(10, 8), font_size=12, label_size=16, save_fig=False, write_path=None*)

Compare the DFTs of the projected residuals with each other.

Calculates the DFT of the residuals in each eigendirection given and plots it alongside the residuals themselves. Produces a separate figure per eigendirection. Use plot_residuals_dft if a single figure with each eigendirection included as a subplot is desired. The length of the time series are padded with zeroes up to the next power of two.

Parameters

- **dates** (*datetime.datetime*) – dates of time series measurements.
- **projected_residuals** (*time series*) – difference between modelled and SV rotated into the eigendirections obtained during denoising (principal component analysis). The proxy for unmodelled external signal is the residual projected in the noisiest eigendirection(s).
- **fig_size** (*array*) – figure size in inches. Defaults to 8 inches by 6 inches.
- **font_size** (*int*) – font size for axes. Defaults to 12 pt.
- **label_size** (*int*) – font size for axis labels. Defaults to 16 pt.
- **plot_legend** (*bool*) – option to include a legend on the plot. Defaults to True.
- **save_fig** (*bool*) – option to save figure. Defaults to False.
- **write_path** (*str*) – output path for figure if saved.

magpysv.plots.**plot_sv**(**, dates, sv, model, obs, model_name, fig_size=(8, 6), font_size=12, label_size=16, plot_legend=False, plot_average=False, window_length=12, min_samples=3, save_fig=False, write_path=None*)

Plot the SV and model prediction for a single observatory.

Produces a plot of the X, Y and Z components of the SV and field model prediction for a single observatory.

Parameters

- **dates** (*datetime.datetime*) – dates of time series measurements.
- **sv** (*time series*) – X, Y and Z components of SV at a single location.

- **model** (*time series*) – X, Y and Z components of the SV predicted by a field model for the same location as the data.
- **obs** (*str*) – observatory name given as three digit IAGA code.
- **model_name** (*str*) – field model name.
- **fig_size** (*array*) – figure size in inches. Defaults to 8 inches by 6 inches.
- **font_size** (*int*) – font size for axes. Defaults to 12 pt.
- **label_size** (*int*) – font size for axis labels. Defaults to 16 pt.
- **plot_legend** (*bool*) – option to include a legend on the plot. Defaults to False.
- **plot_average** (*bool*) – option to include a running average of the SV time series on the plot. Defaults to False.
- **window_length** (*int*) – number of months over which to take the running average if this is plotted. Defaults to 12 months.
- **min_samples** (*int*) – minimum number of non-NaN values that must be present in the window in order for the running average to be calculated rather than set to NaN. Defaults to 3 (e.g. for monthly first differences this means that at least 3 months of data per window are required to calculate the 12-month running average.)
- **save_fig** (*bool*) – option to save figure. Defaults to False.
- **write_path** (*str*) – output path for figure if saved.

magpysv.plots.**plot_sv_comparison**(*, *dates*, *noisy_sv*, *denoised_sv*, *model*, *obs*, *model_name*, *fig_size=(8, 6)*, *font_size=12*, *label_size=16*, *plot_legend=False*, *plot_average=False*, *window_length=12*, *min_samples=3*, *save_fig=False*, *write_path=None*, *corrected_residuals*, *residuals*, *plot_rms=False*)

Plot noisy/denoised SV and model prediction for a single observatory.

Produces a plot of the X, Y and Z components of the noisy SV, the denoised SV and field model prediction for a single observatory.

Parameters

- **dates** (*datetime.datetime*) – dates of time series measurements.
- **noisy_sv** (*time series*) – X, Y and Z components of uncorrected SV at a single location.
- **denoised_sv** (*time series*) – X, Y and Z components of denoised SV at a single location.
- **residuals** (*time series*) – difference between modelled and observed SV.
- **corrected_residuals** (*time series*) – difference between modelled and denoised observed SV.
- **model** (*time series*) – X, Y and Z components of the SV predicted by a field model for the same location as the data.
- **model_name** (*str*) – field model name.
- **obs** (*str*) – observatory name given as three digit IAGA code.
- **fig_size** (*array*) – figure size in inches. Defaults to 8 inches by 6 inches.
- **font_size** (*int*) – font size for axes. Defaults to 12 pt.
- **label_size** (*int*) – font size for axis labels. Defaults to 16 pt.

- **plot_legend** (*bool*) – option to include a legend on the plot. Defaults to False.
- **plot_average** (*bool*) – option to include a running average of the SV time series on the plot. Defaults to False.
- **window_length** (*int*) – number of months over which to take the running average if this is plotted. Defaults to 12 months.
- **min_samples** (*int*) – minimum number of non-NaN values that must be present in the window in order for the running average to be calculated rather than set to NaN. Defaults to 3 (e.g. for monthly first differences this means that at least 3 months of data per window are required to calculate the 12-month running average.)
- **plot_rms** (*bool*) – option to calculate the rms before and after denoising
- **display the values on the figure. Defaults to False.** (*and*) –
- **save_fig** (*bool*) – option to save figure. Defaults to False.
- **write_path** (*str*) – output path for figure if saved.

Returns

tuple containing:

- **rms_ratio_x (*float*):** ratio of rms values for X component residuals before and after denoising.
- **rms_ratio_y (*float*):** ratio of rms values for Y component residuals before and after denoising.
- **rms_ratio_z (*float*):** ratio of rms values for Z component residuals before and after denoising.

Return type (tuple)

magpysv.plots.**rms_ratios**(*, *rms*, *fig_size=(8, 6)*, *font_size=12*, *label_size=16*, *save_fig=False*, *write_path=None*)

Plot the rms of residuals after removing successive eigendirections.

Plots the ratio of the residuals rms values before and after denoising, for different numbers of eigendirections are removed from the data. Removing all eigendirections gives a ratio of zero as the denoised SV now equals the model prediction. Requires the dnoising to be run several times, each time using a different number of eigendirections for the external signal proxy (i.e. different values of the argument proxy_number in calls to eigenvalue_analysis.) Uses output from plot_sv_comparison when that function is run with the option plot_rms=True.

Parameters

- **rms** (*dict*) – rms ratios for each component after running eigenvalue_analysis with different values for the proxy_number argument.
- **fig_size** (*array*) – figure size in inches. Defaults to 8 inches by 6 inches.
- **font_size** (*int*) – font size for axes. Defaults to 12 pt.
- **label_size** (*int*) – font size for axis labels. Defaults to 16 pt.
- **save_fig** (*bool*) – option to save figure. Defaults to False.
- **write_path** (*str*) – output path for figure if saved.

5.6 tools module

Module containing functions to parse files output by magnetic field models.

Part of the MagPySV package for geomagnetic data analysis. This module provides various functions to read SV files output by geomagnetic field models.

magpysv.tools.**apply_Ap_threshold**(*, *Ap_file=None*, *obs_data*, *threshold*)
Remove observatory data for times with ap values above threshold value.

Parameters

- **Ap_file** (*str*) – path to file containing hourly values for the ap index.
- **obs_data** (*pandas.DataFrame*) – dataframe containing hourly means of observed geomagnetic field values.
- **threshold** (*int*) – the threshold Ap value. Data for days with a higher Ap value will be replaced with NaNs and omitted from monthly (or annual etc) means.

Returns data with ap threshold applied.

Return type obs_data (pandas.DataFrame)

magpysv.tools.**calculate_correlation_index**(*, *dates*, *signal*, *index_file*)
Calculate correlation coefficient between signal and a geomagnetic index

Parameters

- **dates** (*datetime.datetime*) – dates of time series measurements.
- **signal** (*time series*) – data to be compared with geomagnetic index.
- **index_file** (*str*) – path to file containing geomagnetic index.

Returns

tuple containing:

- **coeff (*float*):** correlation coefficient.
- **merged (*pandas.DataFrame*):** dataframe containing dates, signal and the geomagnetic index.

Return type (tuple)

magpysv.tools.**calculate_residuals**(*, *obs_data*, *model_data*)
Calculate SV residuals (observed - prediction) using datetime objects.

Parameters

- **obs_data** (*pandas.DataFrame*) – dataframe containing means (usually monthly) of SV calculated from observed geomagnetic field values.
- **model_data** (*pandas.DataFrame*) – dataframe containing the SV predicted by a geomagnetic field model.

Returns dataframe containing SV residuals.

Return type residuals (pandas.DataFrame)

magpysv.tools.**calculate_sv**(*obs_data*, *mean_spacing=1*)
Calculate the secular variation from the observed magnetic field values.

Uses monthly means of geomagnetic observatory data to calculate the SV according to user-specified sampling. The typical choices are monthly differences of monthly means and annual differences of monthly means. For

samplings other than monthly differences, the datetime objects of the calculated SV are taken the midpoint of the datetime objects of the field data. E.g. differencing the means of the field January 1999 and in January 2000 yields the SV for July 1999.

Parameters

- **obs_data** (*pandas.DataFrame*) – dataframe containing means (usually monthly) of observed geomagnetic field values.
- **mean_spacing** (*int*) – the number of months separating the monthly mean values used to calculate the SV. Set to 1 (default) to use adjacent months of data (first differences of monthly means) or set to 12 to calculate annual differences of monthly means.

Returns dataframe containing SV data.

Return type obs_sv (pandas.DataFrame)

magpysv.tools.**calculate_sv_index**(*obs_data*, *mean_spacing=1*)

Calculate the secular variation of a geomagnetic index.

Uses monthly means of a geomagnetic index (e.g. Dst) to calculate the SV according to user-specified sampling. The typical choices are monthly differences of monthly means and annual differences of monthly means. For samplings other than monthly differences, the datetime objects of the calculated SV are taken the midpoint of the datetime objects of the field data. E.g. differencing the means of the field January 1999 and in January 2000 yields the SV for July 1999.

Parameters

- **obs_data** (*pandas.DataFrame*) – dataframe containing means (usually monthly) of observed geomagnetic field values.
- **mean_spacing** (*int*) – the number of months separating the monthly mean values used to calculate the SV. Set to 1 (default) to use adjacent months of data (first differences of monthly means) or set to 12 to calculate annual differences of monthly means.

Returns dataframe containing SV data.

Return type obs_sv (pandas.DataFrame)

magpysv.tools.**correct_baseline_change**(*, *observatory*, *field_data*, *baseline_data*, *print_data*)

Correct documented baseline changes.

Parameters

- **observatory** (*str*) – observatory name given as a 3-digit IAGA code.
- **field_data** (*pandas.DataFrame*) – uncorrected magnetic field data.
- **baseline_data** (*pandas.DataFrame*) – baseline discontinuity data in the format output by get_baseline_info.
- **print_data** (*bool*) – option to print the corrections made.

magpysv.tools.**data_resampling**(*data*, *sampling='MS'*, *average_date=True*)

Resample the hourly geomagnetic data to a specified frequency.

Parameters

- **data** (*pandas.DataFrame*) – dataframe containing datetime objects and hourly means of magnetic data.
- **sampling** (*str*) – new sampling frequency. Default value is 'MS' (monthly means), which averages data for each month and sets the datetime object to the first day of that month. Use 'M' to set the datetime object to the final day of the month. Another useful option is 'A' (annual means), which averages data for a whole year and sets the datetime

object to the final day of the year. Use 'AS' to set the datetime object to the first day of the year.

- **average_date** (*bool*) – the specified resampling intervals only have options for setting the date to the first ('MS' and 'AS') or final ('M' and 'A') day of the month or year. For monthly averages, a more appropriate representative date is the middle of that month (i.e. the 15th day of the month). For annual averages, an appropriate representative date is the middle of that year (taken as July 1st of the year.) This option is used to set the dates to the centre of the sampling interval. Defaults to True.

Returns dataframe of datetime objects and monthly/annual means of observatory data.

Return type data (pandas.DataFrame)

magpysv.tools.**get_baseline_info**(*, *fname=None*)

Read documented baseline changes from a file. :param fname: location of file containing documented baseline changes. :type fname: str

Returns baseline change data.

Return type data (pandas.DataFrame)

magpysv.tools.**remove_selected_points**(*, *data*, *fname*)

Remove specified points from dataset based on list of points in a file.

Reads a list of unwanted points from a file and removes them from the dataframe if present. E.g. If the user had monthly SV means and wished to exclude the X value at NGK in January 2015 from the analysis, the following line would be written in a file

2015-01-01,ngk,X

It is preferable (and more repeatable) to use the included outlier detection algorithms to remove outliers.

Parameters

- **data** (*pandas.DataFrame*) – dataframe containing datetime objects and daily means of magnetic data.
- **fname** (*str*) – path to file containing a list of unwanted data points.

Returns the same dataframe with the points removed.

Return type data (pandas.DataFrame)

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

Index

S

W