PyDIVIDE User's Guide

Bryan Harter
Elysia Lucas

Updated: August 17, 2018

MAVEN Visualization Team
Laboratory for Atmospheric and Space Physics
University of Colorado - Boulder

Contents

1 Introduction

PyDIVIDE is a toolkit that allows the user to quickly plot MAVEN Key Parameter data. This toolkit reads all data into a common structure, which allows comparisons across multiple instruments. PyDIVIDE also includes generic analysis routines for comparing data to existing models of Martian atmosphere.

2 Toolkit Installation

2.1 System Requirements

The MAVEN PyDIVIDE toolkit currently requires Anaconda 5.0 or above. Anaconda will install Python, as well as numerous software libraries for scientific computing. This toolkit is only compatible with Python 3.0 or above.

2.2 Downloading the Toolkit

To install the PyDIVIDE toolkit, type the following command into the local terminal/Anaconda Prompt terminal.

```
>> pip install pydivide
```

The following is necessary for custom bokeh colorbars and timestamps:

```
>> conda install -c bokeh nodejs
```

The PyDIVIDE toolkit can also be downloaded from the MAVEN Science Data Center GitHub page, https://github.com/MAVENSDC. This will require manual installation of all dependencies of PyDIVIDE. It is recommended that PyDIVIDE be installed via the pip command above.

2.3 Updating the Toolkit

The latest version of PyDIVIDE can be installed by typing the following command into the terminal:

```
>> pip install pydivide --upgrade
```

2.4 Mandatory Data Directory Structure

PyDIVIDE requires data files to be stored in an automatically-created directory structure, elaborated upon later in this section. This has a similar format to the SDC and SSL directory structures. The root directory for data storage can be chosen by the user. When first running a `download_files` or `read` procedure, the user will be prompted to select the `root_data_dir`. After the directory is selected, it is saved in `mvn_toolkit_prefs.txt`, and can later be changed manually as desired. After the first selection of the directory, the user will not be prompted by `download_files` or `read` again. `download_files` will place files into the chosen directory structure, and `read` will pull data files from that directory structure. The requisite directory structure is formatted as such:

```
<root_data_dir>/maven/data/sci/kp/insitu/YYYY/MM/
                              /kp/iuvs/YYYY/MM/
```

Level 2 instrument data downloaded via `download_files` will be placed into the following directory structure:

```
<root_data_dir>/maven/data/sci/sta/l2/YYYY/MM/
                              /sep/l2/YYYY/MM/
                              /swi/l2/YYYY/MM/
                              /swe/l2/YYYY/MM/
                              /lpw/l2/YYYY/MM/
                              /mag/l2/YYYY/MM/
                              /iuv/l2/YYYY/MM/
                              /ngi/l2/YYYY/MM/
                              /euv/l2/YYYY/MM/
                              /acc/l2/YYYY/MM/
```

> Note: For Windows systems, the forward slashes above (/) will instead be back slashes (\).

2.5 Starting PyDIVIDE

An IDE is the recommended way to run PyDIVIDE procedures; however, they can also be run from the terminal. To start an interactive session of Python, enter the following commands into the terminal:

```
>> IPython
>> import pydivide
```

PyDIVIDE function calls can now be entered into the terminal.

2.6 Getting More Help

Feel free to express any further problems or questions about installation or operation of the toolkit to the developers at maven_divide@lasp.colorado.edu.

3 Function Categories

This section serves to outline the rough order in which the functions should be called in order to plot/list/model/manipulate various sets of data.

3.1 Downloading and Reading Data Files

`download_files`: Section 4.6
  `read_model_results`: Section 4.14
  `read`: Section 4.13

3.2 Manipulating Key Parameter Data

  `bin`: Section 4.2
  `insitu_search`: Section 4.7
  `resample`: Section 4.15

3.3 Plotting Key Parameter Data

  `altplot`: Section 4.1
  `corona`: Section 4.4
  `map2d`: Section 4.9
  `occultation`: Section 4.10
  `periapse`: Section 4.11
  `plot`: Section 4.12
  `standards`: Section 4.16

3.4 Prediction Models

  `create_model_maps`: Section 4.5
  `interpol_model`: Section 4.8

3.5 Toolkit Utilities

  `cleanup_files`: Section 4.3
  `tplot_varcreate`: Section 4.17

4 PyDIVIDE Functions

All functions within PyDIVIDE are listed in this section in alphabetical order.

4.1 altplot

Function Call:

```
altplot(kp,
        parameter=None,
        time=None,
        sameplot=True,
        list=False,
        title='Altitude Plot',
        qt=True)
```

Description:

Plot the provided data against spacecraft altitude. If time is not provided, plot the entire dataset.

Required Arguments:

`kp`: STRUCT
KP insitu data structure read from file(s).

`parameter`: INT, LIST, STR
Parameter(s) to be plotted. Can be provided as integer (by index) or string (by name: inst.obs). List may contain various data types.

Optional Arguments:

`time`: [STR/INT,STR/INT]
Two-element list of strings or integers indicating the time range to be plotted. Currently, no checks if time range is within data.

`sameplot`: BOOL
If True, put all curves on same axes.
If False, generate new axes for each plot.

`list`: BOOL
List all KP parameters instead of plotting.

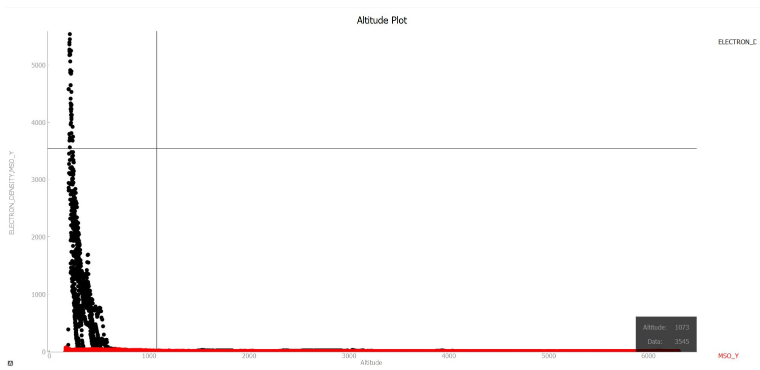
`title`: STR
Sets plot title. Default is ‘Altitude Plot’.

`qt`: BOOL
If True, plot with PyQtGraph.
If False, plot with bokeh.

Examples:

Plot `LPW.ELECTRON_DENSITY` and `MAG.MSO_Y` against spacecraft altitude.

```
>> pydivide.altplot(insitu,
   parameter=['LPW.ELECTRON_DENSITY','MAG.MSO_Y'])
```



4.2 bin

Function Call:

```
bin(kp,
    parameter=None,
    bin_by=None,
    mins=None,
    maxs=None,
    binsize=None,
    avg=False,
    std=False,
    density=False,
    median=False)
```

Description:

Bins insitu Key Parameters by up to 8 different parameters, specified within the data structure. Necessary that at least one of `avg`, `std`, `median`, or `density` be specified.

Required Arguments:

`kp`: STRUCT
  KP insitu data structure read from file(s).

`parameter`: STR
  Key Parameter to be binned. Only one may be binned at a time.

`bin_by`: INT, STR
  Parameters (index or name) by which to bin the specified Key Parameter.

`binsize`: INT, LIST
  Bin size for each binning dimension. Number of elements must be equal to those in `bin_by`.

Optional Arguments:

`mins`: INT, LIST
  Minimum value(s) for each binning scheme. Number of elements must be equal to those in `bin_by`.

`maxs`: INT, LIST

Maximum value(s) for each binning scheme. Number of elements must be equal to those in `bin_by`.

`avg`: BOOL

Calculate average per bin.

`std`: BOOL

Calculate standard deviation per bin.

`density`: BOOL

Returns number of items in each bin.

`median`: BOOL

Calculate median per bin.

Returns:

This procedures outputs up to 4 arrays to user-defined variables, corresponding to `avg`, `std`, `median`, and `density`.

Examples:

Bin STATIC O^+ characteristic energy by spacecraft latitude (1° resolution) and longitude (2° resolution).

```
>> output_avg = pydivide.bin(insitu,
                parameter='static.oplus_char_energy',
                bin_by=['spacecraft.geo_latitude',
                'spacecraft.geo_longitude'],
                avg=True,binsize=[2,1])
```

Bin SWIA H^+ density by spacecraft altitude (10km resolution), return average value and standard deviation for each bin.

```
>> output_avg,output_std = pydivide.bin(insitu,
                           parameter='swia.hplus_density',
                           bin_by='spacecraft.altitude',
                           binsize=10,avg=True,std=True)
```

4.3 cleanup_files

Function Call:

```
cleanup_files()
```

Description:

Searches code directory for *.tab files, keeps latest versions/revisions, asks to delete old versions/revisions. Will ignore files not ending in .tab and not starting with "mvn_kp_insitu" or "mvn_kp_iuvs".

Required Arguments:

None.

Optional Arguments:

None.

File Requirements:

All *.tab files must be named with the following formats:

"mvn_kp_insitu_YYYYMMDD_vXX_rXX.tab"

Ex: mvn_kp_insitu_20170619_v13_r04.tab
"mvn_kp_iuvs_ORBIT_YYYYMMDDTHHMMSS_vXX_rXX.tab"
Ex: mvn_kp_iuvs_02403_20151225T003727_v07_r01.tab

Any extraneous characters or formatting changes to the filename are not compatible with the function regexing.

Examples:

Remove all out-of-date insitu and IUVS files from the local directory.

```
>> pydivide.cleanup_files()
```

4.4 corona

Function Call:

```
corona(iuvs,
       sameplot=True,
       density=True,
       radiance=True,
       orbit_num=None,
       species=None,
       log=False,
       title='IUVS Corona Observations',
       qt=True)
```

Description:

Create altitude plots of corona limb scans from IUVS KP files, containing radiance and density data of various chemical species.

Required Arguments:

`iuvs`: STRUCT
IUVS Key Parameter data structure returned from `read`.

Optional Arguments:

`sameplot`: BOOL
If True, will plot everything on one plot.
If False, will create stacked plots.

`density`: BOOL
Plot density.

`radiance`: BOOL
Plot radiance.

`orbit_num`: INT, LIST
Orbit number(s) to be plotted.

`species`: STR, LIST
Plots only specified chemical species (Appendix A).

`log`: BOOL
Sets logarithmic axes for plotting.

`title`: STR
Sets plot title. Default is 'IUVS Corona Observations'.

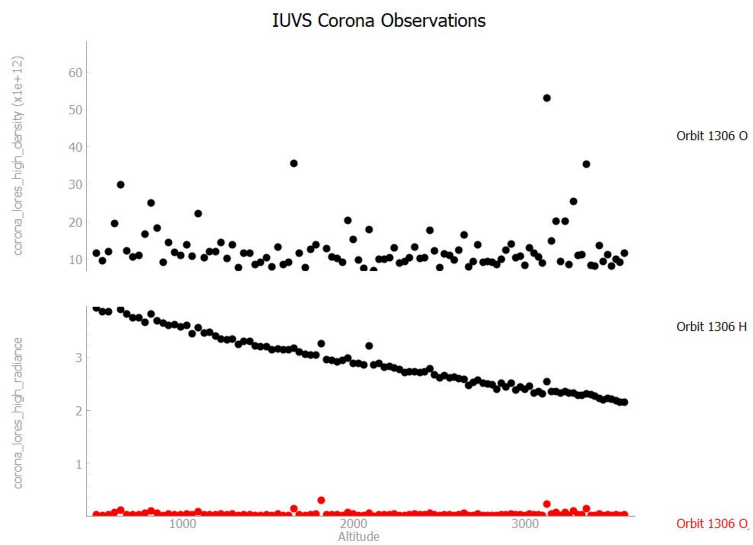
`qt`: BOOL

If True, plot with PyQtGraph.
If False, plot with bokeh.

Examples:

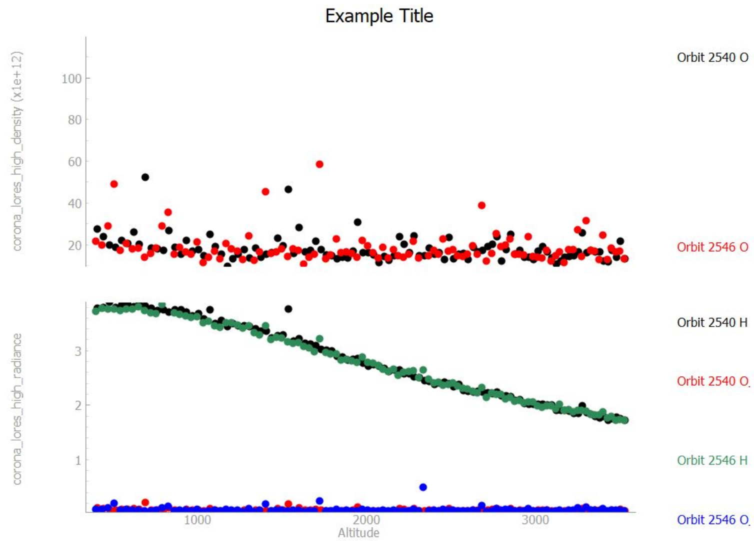
Plot all IUVS radiance and density data.

```
>> insitu,iuvs = pydivide.read('2015-06-02','2015-06-03')
>> pydivide.corona(iuvs)
```



Plot IUVS radiance and density data for H, O, and O_{1304} for orbits 2540 and 2456.

```
>> insitu,iuvs = pydivide.read('2016-01-19','2016-01-20')
>> pydivide.corona(iuvs,
                   species=['H','O','O_1304'],
                   orbit_num=[2540,2456],
                   title='Example Title')
```



4.5 create_model_maps

Function Call:

```
create_model_maps(altitude,
                  variable=None,
                  model=None,
                  file=None,
                  numContours=25,
                  fill=False,
                  ct='viridis',
                  transparency=1,
                  nearest=False,
                  linear=True,
                  saveFig=True)
```

Description:

Generates a .png contour map of a model at a specific altitude. These can be used as a background in `map2d`. The models must be downloaded manually from the SDC website:
https://lasp.colorado.edu/maven/sdc/public/pages/models.html.

Required Arguments:

`altitude`: INT
    Specified altitude of output map.

`variable`: STR
    Plots specified chemical species (Appendix A).

`model`: DICT

Model variable produced from prior call to `read_model_results`.

`file`: STR

If model *not* provided (produced from `read_model_results`), full path to model can be set and read.

Optional Arguments:

`numContours`: INT

Specifies number of contour lines. Default is 25.

`fill`: BOOL

If True, fills in contour levels instead of generating lines.

`ct`: STR

Sets color table. Valid color tables can be found here:
https://matplotlib.org/examples/color/colormaps_reference.html

`transparency`: INT, FLOAT

Sets transparency between [0,1] inclusive. 0 is completely transparent, and 1 is completely opaque.

`nearest`: BOOL

If True, instead of interpolating nearby values, this returns the value of the nearest neighbor altitude.

`linear`: BOOL

If True, performs linear interpolation between 2 altitude layers.

`saveFig`: BOOL

If True, saves figure as .png file.

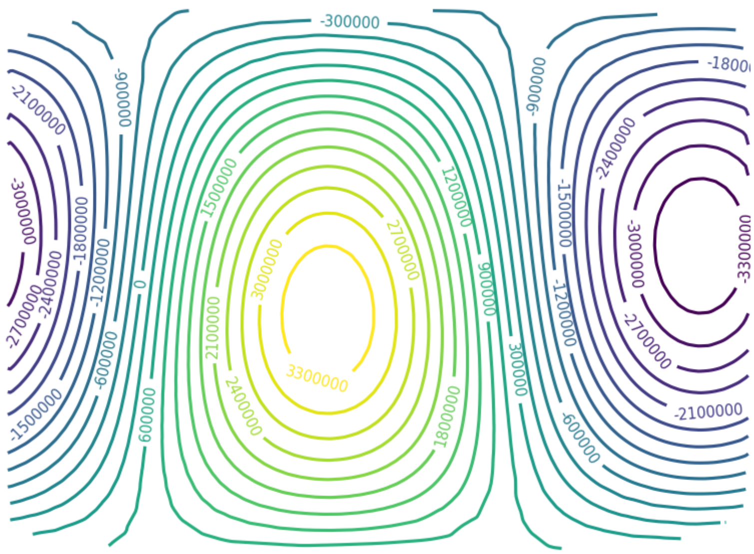
Returns:

A .png file will be created in the same directory as the specified input model.

Examples:

Interpolate all model tracers to spacecraft trajectory using nearest neighbor interpolation.

```
>> pydivide.create_model_maps(altitude=170,
   file = '<dir_path>/MAMPS_LS180_F130_081216.nc',
   variable='geo_x',
   saveFig=True)
```



4.6 download_files

Function Call:

```
download_files(filenames=None,
               instruments=None,
               list_files=False,
               level='l2',
               insitu=True,
               iuvs=False,
               new_files=False,
               start_date='2014-01-01',
               end_date='2020-01-01',
               update_prefs=False,
               only_update_prefs=False,
               exclude_orbit_file=False,
               local_dir=None)
```

Description:

Download insitu and/or IUVS Key Parameter (KP) data files from the MAVEN SDC web server. Also compatible with instrument-specific data downloads. `insitu`, `iuvs`, or at least one `instrument` must be specified.

Required Arguments:

`insitu`: BOOL

Search/download insitu KP data files.

`iuvs`: BOOL

Search/download IUVS KP data files.

`instruments`: STR, LIST

Search/download data for one or more 3-character instrument abbreviations listed in Section 2.4.

Optional Arguments:

`filenames`: STR, ARRAY

Specific filename strings to search/download.

`list_files`: BOOL

List files available for download based on various parameters.

`level`: STR

Specifies desired data level to download, requires specific instrument argument(s). Options include: l0
l1, l1a, l1b, l1c
l2, l2a, l2b, l2c
l3, l3a, l3b, l3c

By default, KP data is downloaded, at which the data is sufficiently calibrated for science analysis.

`new_files`: BOOL

Search/download files on the SDC server that do not exist locally.

`start_date`: 'YYYY-MM-DD'

Search/download data from start_date to present, inclusive.

`end_date`: 'YYYY-MM-DD'

Search/download data prior to end_date, inclusive.

`update_prefs`: BOOL

Before searching/downloading data, open window to allow user to update `root_data_dir` in `mvn_toolkit_prefs.txt`. Once the new path is selected, the search/download will proceed according to the remaining arguments.

`only_update_prefs`: BOOL

Like `update_prefs`, but will not attempt to search/download files.

`exclude_orbit_file`: BOOL

Do not download new version of orbit number file from https://naif.jpl.nasa.gov/naif/.

`local_dir`: STR

Specify a directory for file download. Overrides (but does not overwrite) `root_data_dir` in `mvn_toolkit_prefs.txt`.

Examples:

Download all available insitu data between 2015-01-01 and 2015-01-31, inclusive:

```
>> pydivide.download_files(start_date='2015-01-01',
                           end_date='2015-01-31',
                           insitu=True)
```

List all available CDF insitu KP files on the server:

```
>> pydivide.download_files(insitu=True,
                           list_files=True)
```

Download all *new* IUVS files from 6 April 2015 not found in the local directory.

```
>> pydivide.download_files(iuvs=True,
                            new_files=True,
                            end_date='2015-04-06')
```

List all available Level 2 data files for SWIA.

```
>> pydivide.download_files(instruments='swi',
                            list_files=True,
                            level='l2')
```

List all available Level 2 data files for SWIA for the month of January 2015.

```
>> pydivide.download_files(start_date='2015-01-01',
                            end_date='2015-01-31',
                            instruments='swi',
                            list_files=True,
                            level='l2')
```

Download all *new* Level 2 data files for NGIMS, STATIC, and EUV.

```
>> pydivide.download_files(instruments=['ngi','sta','euv'],
                            new_files=True)
```

4.7 insitu_search

Function Call:

```
insitu_search(kp,
              parameter,
              min=None,
              max=None,
              list=False)
```

Description:

Search existing insitu data structure for specified data, output new structure. containing only the specified data.

Required Arguments:

`kp`: STRUCT

KP insitu data structure read from file(s).

Optional Arguments:

`parameter`: STR, INT, LIST

Name or index of Key Parameter to be searched. May also be entered as list of strings or indices to search for multiple parameters.

`min`: INT, LIST

Minimum value for specified search criteria. If excluded, minimum is assumed to be $-\infty$. If multiple minima specified, they will be applied respectively to specified parameters.

`max`: INT, LIST

Maximum value for specified search criteria. If excluded, maximum is assumed to be ∞. If multiple maxima specified, they will be applied respectively to specified parameters.

`list`: BOOL

Display ordered list of all parameters in `insitu` input. If list is called, no other arguments will be executed, and no output data structure will be returned.

Examples:

Find all STATIC O^+ density data greater than 3000 cm^{-3} and less than 1000000 cm^{-3}, store results in `insitu_new`.

```
>> insitu_new = pydivide.insitu_search(insitu,
                    parameter='static.oplus_density',
                    min=3000,
                    max=1000000)
```

4.8 interpol_model

Function Call:

```
interpol_model(kp,
                    model=None,
                    file=None,
                    nearest=False)
```

Description:

Reads in MAVEN's position from insitu, and determines the value of the models at those points.

Required Arguments:

`kp`: STRUCT

KP insitu data structure read from file(s).

`model`: STR

Source of simulation data to be interpolated.

`file`: STR

If model not provided, can specify the full path to the model.

Optional Arguments:

`nearest`: BOOL

If True, instead of interpolating nearby values, this returns the value of the nearest neighbor altitude.

Returns:

Returns array of data representative of what the spacecraft would have measured if it were traveling through the model.

Examples:

Interpolate all model tracers to the spacecraft trajectory using nearest neighbor interpolation.

```
>> results = pydivide.interpol_model(insitu,
             file='<dir_path>/Elew_18_06_14_t00600.nc',
             nearest=True)
```

4.9 map2d

Function Call:

```
map2d(kp,
      parameter=None,
      time=None,
      list=False,
      color_table=None,
      subsolar=False,
      mso=False,
      map_limit=None,
      basemap=None,
      alpha=None,
      title='MAVEN Mars',
      qt=True)
```

Description:

Produces a 2D map of Mars, either in the planetocentric or MSO coordinate system, with the MAVEN orbital projection and a variety of basemaps. Spacecraft orbital path may be colored by a given insitu KP data value.

Required Arguments:

`kp`: STRUCT
  KP insitu data structure read from file(s).

`parameter`: STR
  Insitu Key Parameter for setting spacecraft trajectory color.

Optional Arguments:

`time`: LIST
  Plots subset of insitu KP data. `time` must be expressed in the format 'YYYY-MM-DD HH:MM:SS'.

`list`: BOOL
  Display list of all parameters in data structure. All other keywords will be ignored if set.

`color_table`: STR, LIST
  Specifies color table(s) to use for plotting.
  `subsolar`: BOOL
  Plot path of subsolar point, not valid for MSO coordinates.

`mso`: BOOL
  Plot using MSO map projection.

`map_limit`: LIST
  Sets the bounding box on the map in lat/lon coordinates: [x0,y0,x1,y1].

`basemap`: STR

Name of basemap on which the spacecraft data will be overlaid. Choices include:

- 'mdim': Mars Digital Image Model
- 'mola': Mars Topography (color)
- 'mola_bw': Mars Topography (black and white)
- 'mag': Mars Crustal Magnetism
- '<dir_path>/file.png': User-defined basemap

`alpha`: INT, FLOAT
    Sets trajectory transparency, valid between [0,1] inclusive.

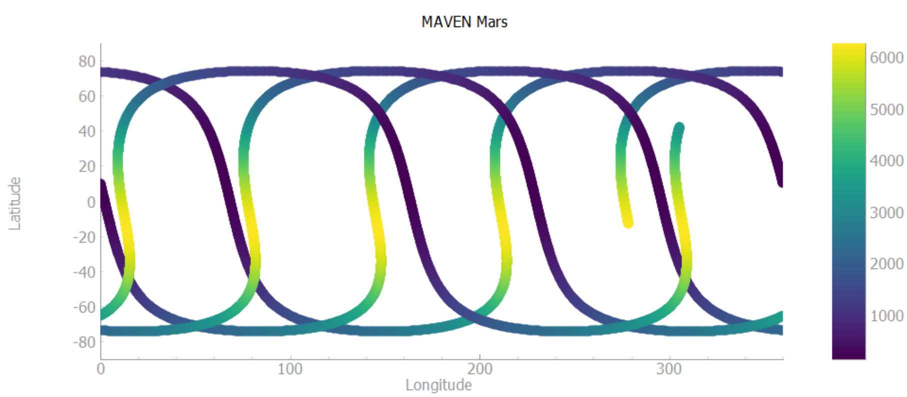
`title`: STR
    Sets plot title.

`qt`: BOOL
    If True, plot with PyQtGraph.
    If False, plot with bokeh.

Examples:

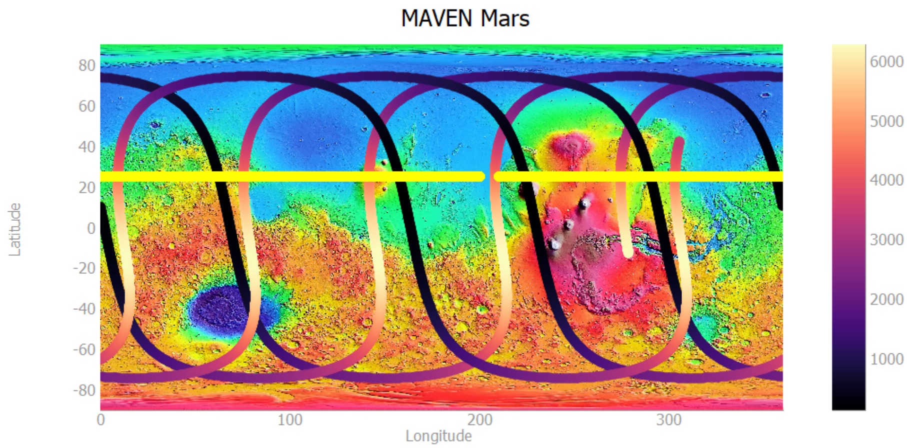
Plot spacecraft altitude along MAVEN surface orbital track.

```
>> pydivide.map2d(insitu,
                  'spacecraft.altitude')
```



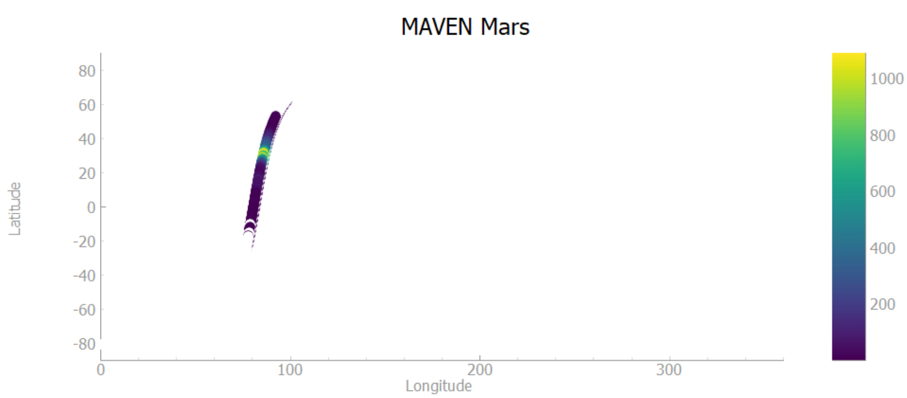
Plot spacecraft altitude along MAVEN surface orbital track using MOLA altimetry basemap; plot subsolar point path.

```
>> pydivide.map2d(insitu,
                  'spacecraft.altitude',
                  basemap='mola',
                  subsolar=True)
```



Plot NGIMS $CO_2{}^+$ density along MAVEN surface orbital track. Limit map to ±60° latitude and 90° to 270° longitude in MSO coordinates.

```
>> pydivide.map2d(insitu,
                  'ngims.co2plus_density',
                  map_limit=[-60,90,60,270],
                  mso=True)
```



4.10 occultation

Function Call:

```
occultation(iuvs,
            sameplot=True,
            orbit_num=None,
            species=None,
            log=False,
            title='IUVS Occultation Observations',
            qt=True):
```

Description:

Plots altitude vs. various species' densities.

Required Arguments:

`iuvs`: STRUCT

IUVS Key Parameter data structure returned from `read`.

Optional Arguments:

`sameplot`: BOOL

If True, will plot everything on one plot.
If False, will create stacked plots.

`orbit_num`: INT, LIST

Orbit number(s) to be plotted.

`species`: STR, LIST

Plots only specified chemical species (Appendix A).

`log`: BOOL

Sets logarithmic axes for plotting.

`title`: STR

Sets plot title. Default is 'IUVS Occultation Observations'.

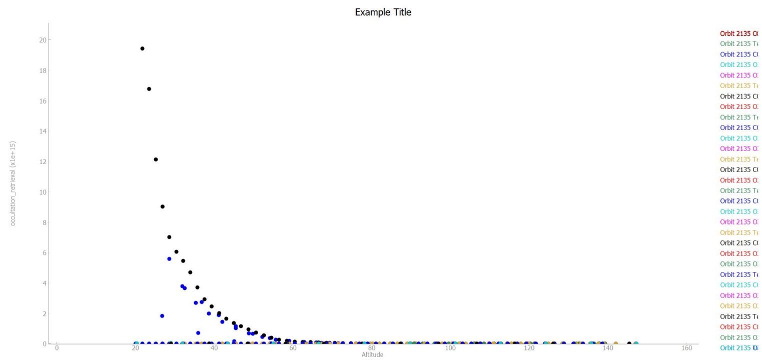
`qt`: BOOL

If True, plot with PyQtGraph.
If False, plot with bokeh.

Examples:

Plot IUVS data for orbit 2135.

```
>> insitu,iuvs = pydivide.read('2015-11-03','2015-11-04')
>> pydivide.occultation(iuvs,
                        orbit_num=[2135],
                        title='Example Title')
```



4.11 periapse

Function Call:

```
periapse(iuvs,
         sameplot=True,
         density=True,
         radiance=True,
         orbit_num=None,
         species=None,
         obs_num=None,
         log=False,
         title='IUVS Periapse Observations',
         qt=True)
```

Description:

Create altitude plots of periapse limb scans from IUVS KP files. These scans contain radiance and density data of various species.

Required Arguments:

`iuvs`: STRUCT

IUVS Key Parameter data structure returned from `read`.

Optional Arguments:

`sameplot`: BOOL

If True, will plot everything on one plot.
If False, will create stacked plots.

`density`: BOOL

Plot density.

`radiance`: BOOL

Plot radiance.

`orbit_num`: INT, LIST

Orbit number(s) to be plotted.

`species`: STR, LIST

Plots only specified chemical species (Appendix A).

`obs_num`: INT, LIST

Observation number(s) to be plotted. There are up to 3 periapse observations per orbit.

`log`: BOOL

Sets logarithmic axes for plotting.

`title`: STR

Sets plot title. Default is 'IUVS Periapse Observations'.

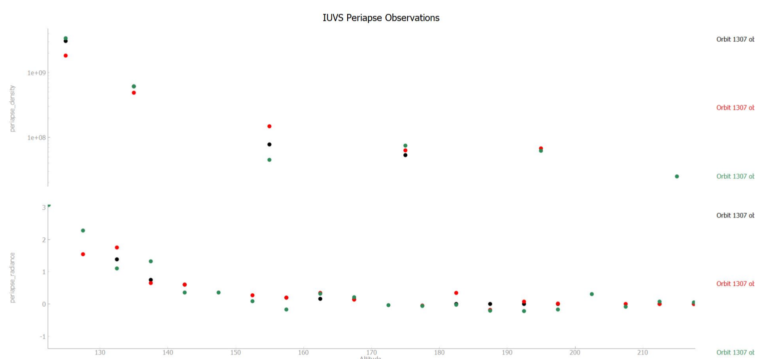
`qt`: BOOL

If True, plot with PyQtGraph.
If False, plot with bokeh.

Examples:

Plot Orbit 1307 N_2 density and radiance data.

```
>> pydivide.periapse(iuvs,
                     log=True,
                     species='N2',
                     orbit_num=1307)
```



4.12 plot

Function Call:

```
plot(kp,
     parameter=None,
     time=None,
     sameplot=True,
     list=False,
     title = '',
     qt=True)
```

Description:

Plot time-series data from `insitu` data structure.

Required Arguments:

`kp`: STRUCT

KP insitu data structure read from file(s).

`parameter`: INT, STR

Name or index of Key Parameter(s) to be plotted.

Optional Arguments:

`time`: [STR,STR]

Plots subset of insitu KP data. `time` must be expressed in the format ['YYYY-MM-DD HH:MM:SS','YYYY-MM-DD HH:MM:SS'].

`sameplot`: BOOL

If True, will plot everything on one plot.
If False, will create stacked plots.

`list`: BOOL

List all KP parameters.

`title`: STR

Sets plot title.

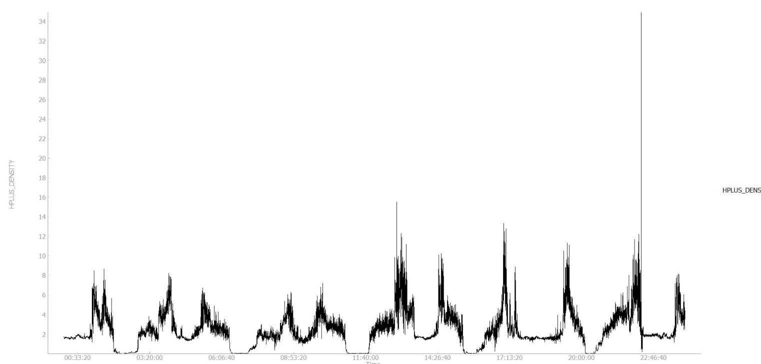
`qt`: BOOL

If True, plot with PyQtGraph.
If False, plot with bokeh.

Examples:

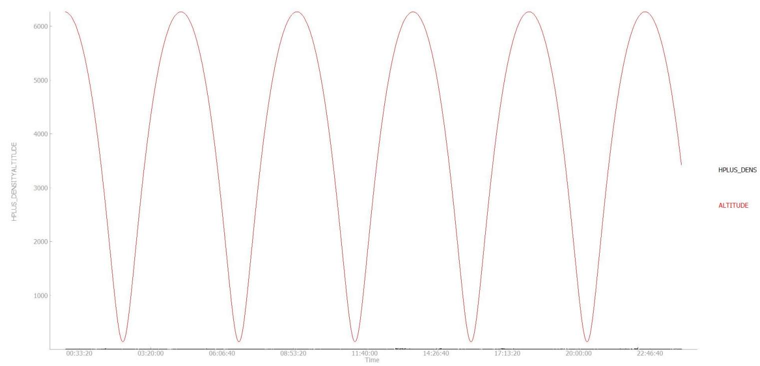
Plot SWIA H^+ density.

```
>> pydivide.plot(insitu,parameter='swia.hplus_density')
```



Plot SWIA H^+ density and altitude in the same window.

```
>> pydivide.plot(insitu,parameter=['swia.hplus_density',
                  'spacecraft.altitude'],sameplot=True)
```

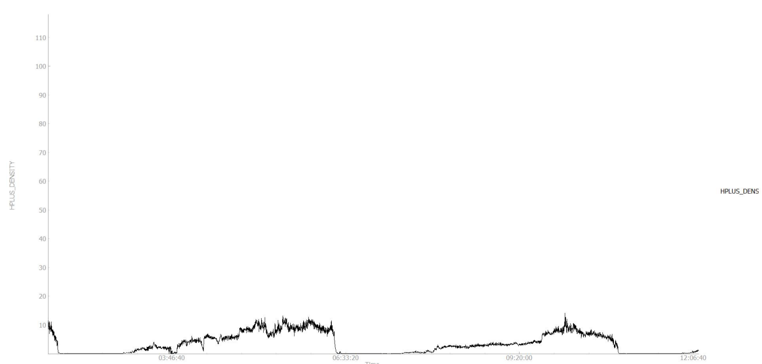


List all KP data parameters.

```
>> pydivide.plot(insitu,list=True)
```

Plot SWIA H^+ density between 02:00 and 12:00 UTC on 2016-04-10.

```
>> pydivide.plot(insitu,parameter='swia.hplus_density',
   time=['2016-04-10 02:00:00','2016-04-10 12:00:00'])
```



4.13 read

Function Call:

```
read(input_time,
     instruments=None,
     insitu_only=False
```

Description:

Ingests a subset of local KP data into one or more data structures. These data structures are the primary inputs to various functions in the PyDIVIDE and PyTplot toolkits. Upon first calling the `read` function, the user will be prompted to choose the `root_data_dir`.

Required Arguments:

`input_time`: DATE, [START DATE, END DATE].

The user must provide a time constraint for data retrieval. These constraints must be provided in either of the following formats:

'YYYY-MM-DD'

'YYYY-MM-DD HH:MM:SS'

For a single time string, the function will return data for the default time period (1 day = 86400 s), beginning at the specified time. The user may also enter a two-element list, corresponding to beginning and end times.

Optional Arguments:

`instruments`: STR, LIST

One or more 3-character instrument abbreviations for instrument-specific data retrieval.

`insitu_only`: BOOL

Will only read insitu instrument data.

Returns:

`insitu`: STRUCT

Contains insitu instrument KP data, as well as spacecraft position and orientation information.

`iuvs`: STRUCT

Contains IUVS instrument KP data.

Examples:

Retrieve insitu and IUVS data for LPW and MAG on 2015-12-26.

```
>> insitu,iuvs = pydivide.read('2015-12-26',
                 instruments=['lpw','mag'])
```

Retrieve only insitu data for all instruments on 2017-06-19.

```
>> insitu = pydivide.read('2017-06-19',
                          insitu_only=True)
```

4.14 read_model_results

Function Call:

```
read_model_results(file)
```

Description:

Reads results of specified simulation into a dictionary object containing sub-directories for metadata, dimension information, and model tracers. This function can read any of the models currently on the MAVEN SDC website with the .nc extension, which can be found here: https://lasp.colorado.edu/maven/sdc/public/pages/models.html The desired model must be downloaded prior to running this procedure.

Required Arguments:

`file`: STR

Simulation result filename to be read.

Optional Arguments:

None.

Returns:

Dictionary object of simulation results, structured roughly as follows:

```
output
     meta
         longsubsol
         ls
         ...
     dim
         lat/x
         lon/y
         alt/z
     var1
         dim_order (x,y,z; z,y,x; etc)
         data
     var2
         dim_order
         data
     ...
```

Examples:

Read the University of Michigan group's ionospheric model for mean solar activity ($F_{10.7} = 130$).

```
>> model = pydivide.read_model_results(
           '<dir_path>/MGITM_LS270_F130_150519.nc')
```

Read the LATMOS group's ionospheric model for solar maximum levels.

```
>> model = pydivide.read_model_results(
           '<dir_path>/Heliosares_Ionos_Ls270_SolMax1_26_01_15.nc')
```

4.15 resample

Function Call:

```
resample(kp,
         time)
```

Description:

Modifies KP structure index to user specified time via interpolation.

Required Arguments:

`kp`: STRUCT

KP insitu data structure read from file(s).

`time`: LIST

Specifies subset of insitu KP data for resampling. `time` must be expressed in the format 'YYYY-MM-DD HH:MM:SS'.

Optional Arguments:

None.

Examples:

Resample insitu time to 2016-06-20 coarse survey 3D file time.

```
>> swi_cdf = cdflib.CDF(
             '<dir_path>/mvn_swi_l2_coarsesvy3d_20160620_v01_r00.cdf')
>> newtime = swi_cdf.varget('time_unix')
>> insitu_resampled = pydivide.resample(insitu,
                                        newtime)
```

4.16 standards

Function Call:

```
standards(kp,
          list_plots=False,
          all_plots=False,
          euv=False,
          mag_mso=False,
          mag_geo=False,
          mag_cone=False,
          mag_dir=False,
          ngims_neutral=False,
          ngims_ions=False,
          eph_angle=False,
          eph_geo=False,
          eph_mso=False,
          swea=False,
          sep_ion=False,
          sep_electron=False,
          wave=False,
          plasma_den=False,
```

```
                plasma_temp=False,
                swia_h_vel=False,
                static_h_vel=False,
                static_o2_vel=False,
                static_flux=False,
                static_energy=False,
                sun_bar=False,
                solar_wind=False,
                ionosphere=False,
                sc_pot=False,
                altitude=False,
                title='Standard Plots',
                qt=True)
```

Description:

Generate all or a subset of 25 standardized plots, created from insitu KP data on the MAVEN SDC website.

Required Arguments:

`kp`: STRUCT

KP insitu data structure read from file(s).

`keyword`: BOOL

At least one or more of the following arguments are required; they may be used in conjunction with one another.

`all_plots`: generate all 25 plots
`euv`: EUV irradiance in each of 3 bands
`mag_mso`: magnetic field, MSO coordinates
`mag_geo`: magnetic field, geographic coordinates
`mag_cone`: magnetic clock and cone angles, MSO coordinates
`mag_dir`: magnetic field, radial/horizontal/northward/eastward components
`ngims_neutral`: neutral atmospheric component densities
`ngims_ions`: ionized atmospheric component densities
`eph_angle`: spacecraft ephemeris information
`eph_geo`: spacecraft position, geographic coordinates
`eph_mso`: spacecraft position, MSO coordinates
`swea`: electron parallel/anti-parallel fluxes
`sep_ion`: ion energy flux
`sep_electron`: electron energy flux
`wave`: electric field wave power
`plasma_den`: plasma density
`plasma_temp`: plasma temperature
`swia_h_vel`: H^+ flow velocity, SWIA MSO coordinates
`static_h_vel`: H^+ flow velocity, STATIC MSO coordinates
`static_o2_vel`: ${O_2}^+$ flow velocity, STATIC MSO coordinates
`static_flux`: H^+/H^{++} and pick-up ion omni-directional flux
`static_energy`: H^+/H^{++} and pick-up ion characteristic energy
`sun_bar`: MAVEN sunlight indicator
`solar_wind`: solar wind dynamic pressure

`ionosphere`: electron spectrum shape parameter
`altitude`: spacecraft altitude
`sc_pot`: spacecraft potential

Optional Arguments:

`list_plots`: BOOL
Display list and description of all available plots.

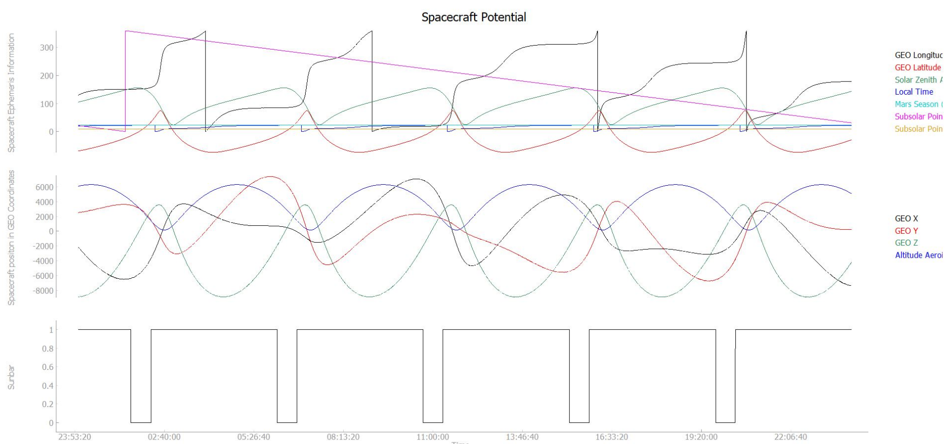
`title`: STR
Title name for all plots.

`qt`: BOOL
If True, plot with PyQtGraph.
If False, plot with bokeh.

Examples:

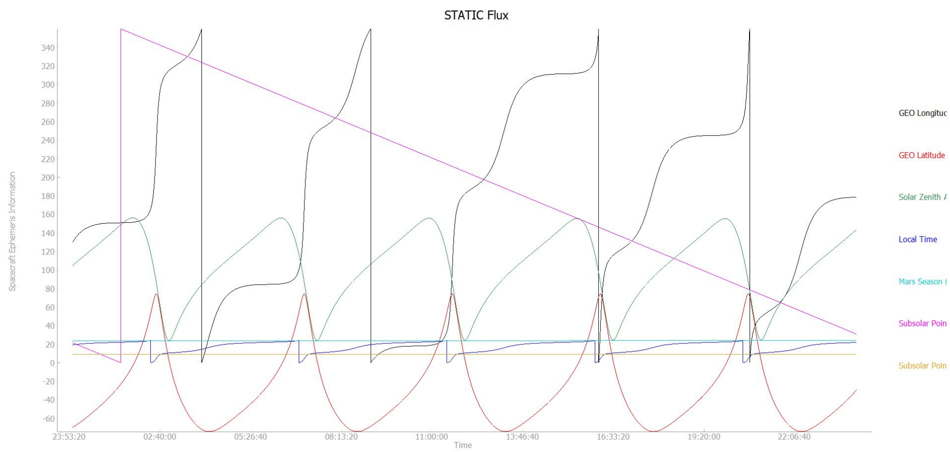
Plot all 25 standard plots. **Note:** not recommended for general use as this will generate 25 very narrow plots; may be useful for a quick glance.

```
>> insitu,iuvs = pydivide.read('2017-06-19','2017-06-20')
>> pydivide.standards(insitu,
                      all_plots=True)
```



Generate figure containing 3 plots: magnetic field standard plot in Mars Solar Orbital coordinates (x, y, z, magnitude), standard spacecraft ephemeris information (sub-spacecraft lat/lon, subsolar lat/lon, local solar time, solar zenith angle, Mars season), and STATIC H^+/H^{++} and pick-up ion omni-directional flux.

```
>> insitu,iuvs = pydivide.read('2017-06-19','2017-06-20')
>> pydivide.standards(insitu,
                      mag_mso=True,
                      eph_angle=True,
                      static_flux=True,
                      title='Example Title')
```



4.17 tplot_varcreate

Function Call:

```
tplot_varcreate(kp)
```

Description:

Creates TVars (tplot variables) for each instrument, named with the format: `mvn_kp::instrument::observation`.

Required Arguments:

`kp`: STRUCT

KP insitu data structure read from file(s).

Optional Arguments:

None.

Examples:

Create TVars for all instruments: EUV, LPW, STATIC, SWEA, SWIA, MAG, SEP, and NGIMS, with data from 2017-06-19.

```
>> insitu = pydivide.read('2017-06-19')
>> pydivide.tplot_varcreate(insitu)
```

A Chemical Species

- C
- C_1561
- C_1657
- CO Cameron
- CO2
- CO2+
- CO2pUVD
- H
- N
- N2
- N_1493
- NO
- O
- O2
- O3
- O_1304
- O_1356
- O_2972

B KP Data Structures

Insitu parameters are callable either by name or number listed below.
IUVS parameters are only callable by name.

B.1 INSITU

1. TimeString
2. Time
3. Orbit
4. IOflag

B.1.1 LPW

5. LPW.ELECTRON_DENSITY
6. LPW.ELECTRON_DENSITY_QUAL_MIN
7. LPW.ELECTRON_DENSITY_QUAL_MAX
8. LPW.ELECTRON_TEMPERATURE
9. LPW.ELECTRON_TEMPERATURE_QUAL_MIN
10. LPW.ELECTRON_TEMPERATURE_QUAL_MAX
11. LPW.SPACECRAFT_POTENTIAL
12. LPW.SPACECRAFT_POTENTIAL_QUAL_MIN
13. LPW.SPACECRAFT_POTENTIAL_QUAL_MAX
14. LPW.EWAVE_LOW_FREQ
15. LPW.EWAVE_LOW_FREQ_QUAL
16. LPW.EWAVE_MID_FREQ
17. LPW.EWAVE_MID_FREQ_QUAL
18. LPW.EWAVE_HIGH_FREQ
19. LPW.EWAVE_HIGH_FREQ_QUAL

B.1.2 EUV

20. EUV.IRRADIANCE_LOW
21. EUV.IRRADIANCE_LOW_QUAL
22. EUV.IRRADIANCE_MID
23. EUV.IRRADIANCE_MID_QUAL
24. EUV.IRRADIANCE_LYMAN
25. EUV.IRRADIANCE_LYMAN_QUAL

B.1.3 SWEA

26. SWEA.SOLAR_WIND_ELECTRON_DENSITY
27. SWEA.SOLAR_WIND_ELECTRON_DENSITY_QUAL
28. SWEA.SOLAR_WIND_ELECTRON_TEMPERATURE
29. SWEA.SOLAR_WIND_ELECTRON_TEMPERATURE_QUAL
30. SWEA.ELECTRON_PARALLEL_FLUX_LOW
31. SWEA.ELECTRON_PARALLEL_FLUX_LOW_QUAL
32. SWEA.ELECTRON_PARALLEL_FLUX_MID
33. SWEA.ELECTRON_PARALLEL_FLUX_MID_QUAL
34. SWEA.ELECTRON_PARALLEL_FLUX_HIGH
35. SWEA.ELECTRON_PARALLEL_FLUX_HIGH_QUAL
36. SWEA.ELECTRON_ANTI_PARALLEL_FLUX_LOW
37. SWEA.ELECTRON_ANTI_PARALLEL_FLUX_LOW_QUAL
38. SWEA.ELECTRON_ANTI_PARALLEL_FLUX_MID
39. SWEA.ELECTRON_ANTI_PARALLEL_FLUX_MID_QUAL
40. SWEA.ELECTRON_ANTI_PARALLEL_FLUX_HIGH
41. SWEA.ELECTRON_ANTI_PARALLEL_FLUX_HIGH_QUAL
42. SWEA.ELECTRON_SPECTRUM_SHAPE_PARAMETER
43. SWEA.ELECTRON_SPECTRUM_SHAPE_PARAMETER_QUAL

B.1.4 SWIA

44. SWIA.HPLUS_DENSITY
45. SWIA.HPLUS_DENSITY_QUAL
46. SWIA.HPLUS_FLOW_VELOCITY_MSO_X
47. SWIA.HPLUS_FLOW_VELOCITY_MSO_X_QUAL
48. SWIA.HPLUS_FLOW_VELOCITY_MSO_Y
49. SWIA.HPLUS_FLOW_VELOCITY_MSO_Y_QUAL
50. SWIA.HPLUS_FLOW_VELOCITY_MSO_Z

51. SWIA.HPLUS_FLOW_VELOCITY_MSO_Z_QUAL
52. SWIA.HPLUS_TEMPERATURE
53. SWIA.HPLUS_TEMPERATURE_QUAL
54. SWIA.SOLAR_WIND_DYNAMIC_PRESSURE
55. SWIA.SOLAR_WIND_DYNAMIC_PRESSURE_QUAL

B.1.5 STATIC

56. STATIC.STATIC_QUALITY_FLAG
57. STATIC.HPLUS_DENSITY
58. STATIC.HPLUS_DENSITY_QUAL
59. STATIC.OPLUS_DENSITY
60. STATIC.OPLUS_DENSITY_QUAL
61. STATIC.O2PLUS_DENSITY
62. STATIC.O2PLUS_DENSITY_QUAL
63. STATIC.HPLUS_TEMPERATURE
64. STATIC.HPLUS_TEMPERATURE_QUAL
65. STATIC.OPLUS_TEMPERATURE
66. STATIC.OPLUS_TEMPERATURE_QUAL
67. STATIC.O2PLUS_TEMPERATURE
68. STATIC.O2PLUS_TEMPERATURE_QUAL
69. STATIC.O2PLUS_FLOW_VELOCITY_MAVEN_APP_X
70. STATIC.O2PLUS_FLOW_VELOCITY_MAVEN_APP_X_QUAL
71. STATIC.O2PLUS_FLOW_VELOCITY_MAVEN_APP_Y
72. STATIC.O2PLUS_FLOW_VELOCITY_MAVEN_APP_Y_QUAL
73. STATIC.O2PLUS_FLOW_VELOCITY_MAVEN_APP_Zz
74. STATIC.O2PLUS_FLOW_VELOCITY_MAVEN_APP_Z_QUAL
75. STATIC.O2PLUS_FLOW_VELOCITY_MSO_X
76. STATIC.O2PLUS_FLOW_VELOCITY_MSO_X_QUAL

77. STATIC.O2PLUS_FLOW_VELOCITY_MSO_Y
78. STATIC.O2PLUS_FLOW_VELOCITY_MSO_Y_QUAL
79. STATIC.O2PLUS_FLOW_VELOCITY_MSO_Z
80. STATIC.O2PLUS_FLOW_VELOCITY_MSO_Z_QUAL
81. STATIC.HPLUS_OMNI_DIRECTIONAL_FLUX
82. STATIC.HPLUS_CHARACTERISTIC_ENERGY
83. STATIC.HPLUS_CHARACTERISTIC_ENERGY_QUAL
84. STATIC.HEPLUS_OMNI_DIRECTIONAL_FLUX
85. STATIC.HEPLUS_CHARACTERISTIC_ENERGY
86. HSTATIC.HEPLUS_CHARACTERISTIC_ENERGY_QUAL
87. STATIC.OPLUS_OMNI_DIRECTIONAL_FLUX
88. STATIC.OPLUS_CHARACTERISTIC_ENERGY
89. STATIC.OPLUS_CHARACTERISTIC_ENERGY_QUAL
90. STATIC.O2PLUS_OMNI_DIRECTIONAL_FLUX
91. STATIC.O2PLUS_CHARACTERISTIC_ENERGY
92. STATIC.O2PLUS_CHARACTERISTIC_ENERGY_QUAL
93. STATIC.HPLUS_CHARACTERISTIC_DIRECTION_MSO_X
94. STATIC.HPLUS_CHARACTERISTIC_DIRECTION_MSO_Y
95. STATIC.HPLUS_CHARACTERISTIC_DIRECTION_MSO_Z
96. STATIC.HPLUS_CHARACTERISTIC_ANGULAR_WIDTH
97. STATIC.HPLUS_CHARACTERISTIC_ANGULAR_WIDTH_QUAL
98. STATIC.DOMINANT_PICKUP_ION_CHARACTERISTIC_DIRECTION_MSO_X
99. STATIC.DOMINANT_PICKUP_ION_CHARACTERISTIC_DIRECTION_MSO_Y
100. STATIC.DOMINANT_PICKUP_ION_CHARACTERISTIC_DIRECTION_MSO_Z
101. STATIC.DOMINANT_PICKUP_ION_CHARACTERISTIC_ANGULAR_WIDTH
102. STATIC.DOMINANT_PICKUP_ION_CHARACTERISTIC_ANGULAR_WIDTH_QUAL

B.1.6 SEP

103. SEP.ION_ENERGY_FLUX__FOV_1_F
104. SEP.ION_ENERGY_FLUX__FOV_1_F_QUAL
105. SEP.ION_ENERGY_FLUX__FOV_1_R
106. SEP.ION_ENERGY_FLUX__FOV_1_R_QUAL
107. SEP.ION_ENERGY_FLUX__FOV_2_F
108. SEP.ION_ENERGY_FLUX__FOV_2_F_QUAL
109. SEP.ION_ENERGY_FLUX__FOV_2_R
110. SEP.ION_ENERGY_FLUX__FOV_2_R_QUAL
111. SEP.ELECTRON_ENERGY_FLUX___FOV_1_F
112. SEP.ELECTRON_ENERGY_FLUX___FOV_1_F_QUAL
113. SEP.ELECTRON_ENERGY_FLUX___FOV_1_R
114. SEP.ELECTRON_ENERGY_FLUX___FOV_1_R_QUAL
115. SEP.ELECTRON_ENERGY_FLUX___FOV_2_F
116. SEP.ELECTRON_ENERGY_FLUX___FOV_2_F_QUAL
117. SEP.ELECTRON_ENERGY_FLUX___FOV_2_R
118. SEP.ELECTRON_ENERGY_FLUX___FOV_2_R_QUAL
119. SEP.LOOK_DIRECTION_1_F_MSO_X
120. SEP.LOOK_DIRECTION_1_F_MSO_Y
121. SEP.LOOK_DIRECTION_1_F_MSO_Z
122. SEP.LOOK_DIRECTION_1_R_MSO_X
123. SEP.LOOK_DIRECTION_1_R_MSO_Y
124. SEP.LOOK_DIRECTION_1_R_MSO_Z
125. SEP.LOOK_DIRECTION_2_F_MSO_X
126. SEP.LOOK_DIRECTION_2_F_MSO_Y
127. SEP.LOOK_DIRECTION_2_F_MSO_Z
128. SEP.LOOK_DIRECTION_2_R_MSO_X
129. SEP.LOOK_DIRECTION_2_R_MSO_Y
130. SEP.LOOK_DIRECTION_2_R_MSO_Z

B.1.7 MAG

131. MAG.MSO_X
132. MAG.MSO_X_QUAL
133. MAG.MSO_Y
134. MAG.MSO_Y_QUAL
135. MAG.MSO_Z
136. MAG.MSO_Z_QUAL
137. MAG.GEO_X
138. MAG.GEO_X_QUAL
139. MAG.GEO_Y
140. MAG.GEO_Y_QUAL
141. MAG.GEO_Z
142. MAG.GEO_Z_QUAL
143. MAG.RMS_DEVIATION
144. MAG.RMS_DEVIATION_QUAL

B.1.8 NGIMS

145. NGIMS.HE_DENSITY
146. NGIMS.HE_DENSITY_PRECISION
147. NGIMS.HE_DENSITY_QUAL
148. NGIMS.O_DENSITY
149. NGIMS.O_DENSITY_PRECISION
150. NGIMS.O_DENSITY_QUAL
151. NGIMS.CO_DENSITY
152. NGIMS.CO_DENSITY_PRECISION
153. NGIMS.CO_DENSITY_QUAL
154. NGIMS.N2_DENSITY
155. NGIMS.N2_DENSITY_PRECISION

156. NGIMS.N2_DENSITY_QUAL
157. NGIMS.NO_DENSITY
158. NGIMS.NO_DENSITY_PRECISION
159. NGIMS.NO_DENSITY_QUAL
160. NGIMS.AR_DENSITY
161. NGIMS.AR_DENSITY_PRECISION
162. NGIMS.AR_DENSITY_QUAL
163. NGIMS.CO2_DENSITY
164. NGIMS.CO2_DENSITY_PRECISION
165. NGIMS.CO2_DENSITY_QUAL
166. NGIMS.O2PLUS_DENSITY
167. NGIMS.O2PLUS_DENSITY_PRECISION
168. NGIMS.O2PLUS_DENSITY_QUAL
169. NGIMS.CO2PLUS_DENSITY
170. NGIMS.CO2PLUS_DENSITY_PRECISION
171. NGIMS.CO2PLUS_DENSITY_QUAL
172. NGIMS.NOPLUS_DENSITY
173. NGIMS.NOPLUS_DENSITY_PRECISION
174. NGIMS.NOPLUS_DENSITY_QUAL
175. NGIMS.OPLUS_DENSITY
176. NGIMS.OPLUS_DENSITY_PRECISION
177. NGIMS.OPLUS_DENSITY_QUAL
178. NGIMS.CO2PLUS_N2PLUS_DENSITY
179. NGIMS.CO2PLUS_N2PLUS_DENSITY_PRECISION
180. NGIMS.CO2PLUS_N2PLUS_DENSITY_QUAL
181. NGIMS.CPLUS_DENSITY
182. NGIMS.CPLUS_DENSITY_PRECISION
183. NGIMS.CPLUS_DENSITY_QUAL

184. NGIMS.OHPLUS_DENSITY
185. NGIMS.OHPLUS_DENSITY_PRECISION
186. NGIMS.OHPLUS_DENSITY_QUAL
187. NGIMS.NPLUS_DENSITY
188. NGIMS.NPLUS_DENSITY_PRECISION
189. NGIMS.NPLUS_DENSITY_QUAL

B.1.9 APP

190. APP.ATTITUDE_GEO_X
191. APP.ATTITUDE_GEO_Y
192. APP.ATTITUDE_GEO_Z
193. APP.ATTITUDE_MSO_X
194. APP.ATTITUDE_MSO_Y
195. APP.ATTITUDE_MSO_Z

B.1.10 SPACECRAFT

196. SPACECRAFT.GEO_X
197. SPACECRAFT.GEO_Y
198. SPACECRAFT.GEO_Z
199. SPACECRAFT.MSO_X
200. SPACECRAFT.MSO_Y
201. SPACECRAFT.MSO_Z
202. SPACECRAFT.SUB_SC_LONGITUDE
203. SPACECRAFT.SUB_SC_LATITUDE
204. SPACECRAFT.SZA
205. SPACECRAFT.LOCAL_TIME
206. SPACECRAFT.ALTITUDE
207. SPACECRAFT.ATTITUDE_GEO_X

208. SPACECRAFT.ATTITUDE_GEO_Y
209. SPACECRAFT.ATTITUDE_GEO_Z
210. SPACECRAFT.ATTITUDE_MSO_X
211. SPACECRAFT.ATTITUDE_MSO_Y
212. SPACECRAFT.ATTITUDE_MSO_Z
213. SPACECRAFT.MARS_SEASON
214. SPACECRAFT.MARS_SUN_DISTANCE
215. SPACECRAFT.SUBSOLAR_POINT_GEO_LONGITUDE
216. SPACECRAFT.SUBSOLAR_POINT_GEO_LATITUDE
217. SPACECRAFT.SUBMARS_POINT_SOLAR_LONGITUDE
218. SPACECRAFT.SUBMARS_POINT_SOLAR_LATITUDE
219. SPACECRAFT.T11
220. SPACECRAFT.T12
221. SPACECRAFT.T13
222. SPACECRAFT.T21
223. SPACECRAFT.T22
224. SPACECRAFT.T23
225. SPACECRAFT.T31
226. SPACECRAFT.T32
227. SPACECRAFT.T33
228. SPACECRAFT.SPACECRAFT_T11
229. SPACECRAFT.SPACECRAFT_T12
230. SPACECRAFT.SPACECRAFT_T13
231. SPACECRAFT.SPACECRAFT_T21
232. SPACECRAFT.SPACECRAFT_T22
233. SPACECRAFT.SPACECRAFT_T23
234. SPACECRAFT.SPACECRAFT_T31
235. SPACECRAFT.SPACECRAFT_T32
236. SPACECRAFT.SPACECRAFT_T33

B.2 IUVS

B.2.1 ALL

- TIME_START
- TIME_STOP
- SZA
- LOCAL_TIME
- LAT
- LON
- LAT_MSO
- LON_MSO
- ORBIT_NUMBER
- MARS_SEASON_LS
- SPACECRAFT_GEO
- SPACECRAFT_MSO
- SUN_GEO
- SPACECRAFT_GEO_LONGITUDE
- SPACECRAFT_GEO_LATITUDE
- SPACECRAFT_MSO_LONGITUDE
- SPACECRAFT_MSO_LATITUDE
- SUBSOLAR_POINT_GEO_LONGITUDE
- SUBSOLAR_POINT_GEO_LATITUDE
- SPACECRAFT_SZA
- SPACECRAFT_LOCAL_TIME
- SPACECRAFT_ALTITUDE
- MARS_SUN_DISTANCE

B.2.2 PERIAPSE1/PERIAPSE2/PERIAPSE3

- SCALE_HEIGHT
- DENSITY
- RADIANCE
- TEMPERATURE
- ALT

B.2.3 APOAPSE

- OZONE_DEPTH
- AURORAL_INDEX
- DUST_DEPTH
- RADIANCE
- SZA_BP
- LOCAL_TIME_BP
- LON_BINS
- LAT_BINS

B.2.4 CORONA_LORES_HIGH

- HALF_INT_DISTANCE
- TEMPERATURE
- DENSITY
- RADIANCE
- ALT

B.2.5 OCCULTATION

- CO2
- O2
- O3
- TEMPERATURE