myDARN Documentation

B. N. Tobin and T. K. Yeoman

Planetary Science Group

University of Leicester

December 2, 2025

Contents

1 Introduction

myDARN is an open-source Python library designed for processing and visualizing SuperDARN data. It extends the functionality of the pyDARN [1] and pyDARNio [2] packages, and additionally provides interfaces for running Radar Software Toolkit (RST) [3] subprocesses directly from Python.

The primary data format used by SuperDARN is the Data Map (DMAP) format, developed by Rob Barnes [1]. This is a binary, self-describing format in which each record contains both metadata and associated data fields. For more detailed information on the DMAP format, refer to the RST documentation [3]. The library currently supports the reading and writing of the following DMAP file types:

- RAWACF
- FITACF
- GRID

myDARN also includes utilities for compressing and decompressing `.bz2` and `.gz` file formats. The supported RST subprocesses in myDARN include `make_fit`, `make_grid`, and `fit_speck_removal`. myDARN also provides several of plotting tools, including one-dimensional and two-dimensional histograms, fan plots over a specified time range, range–time parameter (RTP) plots for scalar and vector field quantities, and scatter plots.

The library source code can be found on the myDARN GitHub repository. If you have any questions or concerns please submit an issue on the myDARN repository.

2 Installation

For most users, myDARN can be installed simply by downloading the source code from the myDARN GitHub repository, navigating to the directory containing the source code, and running:

```
cd path/to/myDARN
pip3 install mydarn
```

If you have multiple Python versions installed, or prefer to keep dependencies isolated, it is recommended to use a virtual environment. For example, creating a pip virtual environment:

```
python3 -m pip install --user virtualenv
python3 -m virtualenv <environment name>
source <environment name>/bin/activate
pip3 install mydarn
```

Alternatively, a conda virtual environment can be created using Anaconda:

```
conda create -n <environment name> python=3.12 anaconda
conda activate <environment name>
pip install mydarn
```

myDARN requires Python 3.7 or later. To check your Python version, run:

```
python --version
```

To use the RST methods provided by myDARN, RST version 5.1 or later must be installed locally. Installation instructions for RST can be found in its official documentation [3].

The following python package dependencies are required:

- pyDARN (version 4.1.2+)
- pyDARNio (version 2.0+)
- matplotlib (version 3.3.4+)
- NumPy

The following modules from the Python standard library are also required and are included with every Python installation (no additional installation required):

- datetime
- glob
- collections
- os
- subprocess
- bz2
- gzip
- copy

3 Data Processing

3.1 Concatenate

Provides methods for reading, writing, and grouping SuperDARN DMAP-based data formats, including FITACF, RAWACF, and GRID files.

Methods:

- `read_fitacf`
  Read and concatenate SuperDARN FITACF data files.
- `read_rawacf`
  Read and concatenate SuperDARN RAWACF data files.
- `read_grid`
  Read and concatenate SuperDARN GRID data files.
- `read_dmap`
  Generic DMAP file reader.
- `write_fitacf`
  Write SuperDARN FITACF data to file.
- `write_rawacf`
  Write SuperDARN RAWACF data to file.
- `write_grid`
  Write SuperDARN GRID data to file.
- `write_dmap`
  Generic DMAP file writer.
- `group_files`
  Group SuperDARN data files by day, month, or year.

Dependencies:

- pyDARN
- pyDARNio
- glob
- collections
- os

3.1.1 read_fitacf

Reads and concatenates SuperDARN FITACF data from one or more files.

Parameters:

- `filepath`: str or list of str
  If provided as a string, it is interpreted as a file path pattern to FITACF SuperDARN data files containing `'*'` wildcard characters. If provided as a list, it must contain explicit file paths to the SuperDARN FITACF files to be processed.

- `print_console`: bool
  If True, progress and status messages are printed to the console.
  Default: True

Returns:

- `data`: list of dict
  A list of FITACF records concatenated from the provided files.

Raises:

- `FileNotFoundError`
- `TypeError`

Sample Code:

```
import mydarn

fitacf_data = mydarn.Concatenate.read_fitacf('file/*.fitacf')
```

3.1.2 read_rawacf

Reads and concatenates SuperDARN RAWACF data from one or more files.

Parameters:

- `filepath`: str or list of str
  If provided as a string, it is interpreted as a file path pattern to RAWACF SuperDARN data files containing `'*'` wildcard characters. If provided as a list, it must contain explicit file paths to the SuperDARN RAWACF files to be processed.
- `print_console`: bool
  If True, progress and status messages are printed to the console.
  Default: True

Returns:

- `data`: list of dict
  A list of RAWACF records concatenated from the provided files.

Raises:

- `FileNotFoundError`
- `TypeError`

Sample Code:

```
import mydarn

rawacf_data = mydarn.Concatenate.read_rawacf('file/*.rawacf')
```

3.1.3 read_grid

Reads and concatenates SuperDARN GRID data from one or more files.

Parameters:

- `filepath`: str or list of str
  If provided as a string, it is interpreted as a file path pattern to GRID SuperDARN data files containing `'*'` wildcard characters. If provided as a list, it must contain explicit file paths to the SuperDARN GRID files to be processed.

- `print_console`: bool
  If True, progress and status messages are printed to the console.
  Default: True

Returns:

- `data`: list of dict
  A list of GRID records concatenated from the provided files.

Raises:

- `FileNotFoundError`

- `TypeError`

Sample Code:

```
import mydarn

grid_data = mydarn.Concatenate.read_grid('file/*.grid')
```

3.1.4 read_dmap

Reads and concatenates generic SuperDARN DMAP data from one or more files.

Parameters:

- `filepath`: str or list of str
  If provided as a string, it is interpreted as a file path pattern to DMAP SuperDARN data files containing `'*'` wildcard characters. If provided as a list, it must contain explicit file paths to the SuperDARN DMAP files to be processed.

- `print_console`: bool
  If True, progress and status messages are printed to the console.
  Default: True

Returns:

- `data`: list of dict
  A list of DMAP records concatenated from the provided files.

Raises:

- `FileNotFoundError`

- `TypeError`

Sample Code:

```
import mydarn

dmap_data = mydarn.Concatenate.read_dmap('file/*.fitacf')
```

3.1.5 write_fitacf

Writes SuperDARN FITACF data to a FITACF file.

Parameters:

- `data`: list of dict
  A list of SuperDARN FITACF records to be written.
- `filepath`: str
  The output file path, including the desired filename.
- `print_console`: bool, optional
  If True, progress and status messages are printed to the console.
  Default: True

Returns:

- `None`
  This function does not return any value.

Raises:

- `TypeError`

Sample Code:

```
import mydarn

fitacf_data = mydarn.Concatenate.read_fitacf('file/*.fitacf')
mydarn.Concatenate.write_fitacf(fitacf_data, 'file/new_data_file.fitacf')
```

3.1.6 write_rawacf

Writes SuperDARN RAWACF data to a RAWACF file.

Parameters:

- `data`: list of dict
  A list of SuperDARN RAWACF records to be written.
- `filepath`: str
  The output file path, including the desired filename.
- `print_console`: bool, optional
  If True, progress and status messages are printed to the console.
  Default: True

Returns:

- `None`
  This function does not return any value.

Raises:

- `TypeError`

Sample Code:

```
import mydarn

rawacf_data = mydarn.Concatenate.read_rawacf('file/*.rawacf')
mydarn.Concatenate.write_rawacf(rawacf_data, 'file/new_data_file.rawacf')
```

3.1.7 write_grid

Writes SuperDARN GRID data to a GRID file.

Parameters:

- `data`: list of dict
  A list of SuperDARN GRID records to be written.
- `filepath`: str
  The output file path, including the desired filename.
- `print_console`: bool, optional
  If True, progress and status messages are printed to the console.
  Default: True

Returns:

- `None`
  This function does not return any value.

Raises:

- `TypeError`

Sample Code:

```
import mydarn

grid_data = mydarn.Concatenate.read_grid('file/*.grid')
mydarn.Concatenate.write_grid(grid_data, 'file/new_data_file.grid')
```

3.1.8 write_dmap

Writes generic SuperDARN DMAP data to a DMAP file.

Parameters:

- `data`: list of dict
  A list of SuperDARN DMAP records to be written.

- `filepath`: str
  The output file path, including the desired filename.
- `print_console`: bool, optional
  If True, progress and status messages are printed to the console.
  Default: True

Returns:

- `None`
  This function does not return any value.

Raises:

- `TypeError`

Sample Code:

```
import mydarn

dmap_data = mydarn.Concatenate.read_dmap('file/*.fitacf')
mydarn.Concatenate.write_dmap(dmap_data, 'file/new_data_file.fitacf')
```

3.1.9 group_files

Groups SuperDARN data files based on date information.

Parameters:

- `filepath`: str or list of str
  If provided as a string, it is interpreted as a file path pattern to SuperDARN data files containing `'*'` wildcard characters. If provided as a list, it must contain explicit file paths to the SuperDARN files to be processed.
- `group_by`: str, optional
  Specifies how the files should be grouped: `'day'`, `'month'`, or `'year'`.
  Default: `'day'`

Returns:

- `groups`: dict
  A dictionary where each key is a date (formatted according to the grouping method), and each value is a list of file paths corresponding to that date.

Raises:

- `FileNotFoundError`
- `NameError`
- `TypeError`

Sample Code:

```
import mydarn

fitacf_files = ['file/20250101.fitacf', 'file/20250109.fitacf',
↪  'file/20250216.fitacf', 'file/20250122.fitacf', 'file/20250202.fitacf']

grouped_files = mydarn.Concatenate.group_files(fitacf_files, 'month')
print(grouped_files)
```

Output:

```
{('2025', '01'): ['file/20250101.fitacf',
                  'file/20250109.fitacf',
                  'file/20250122.fitacf'],
 ('2025', '02'): ['file/20250202.fitacf',
                  'file/20250216.fitacf']}
```

3.2 Convert

Provides methods for compressing and uncompressing `.gz` and `.bz2` files.

Methods:

- `unzip_bz2`
  Uncompress `.bz2` files and remove the original archives.
- `zip_bz2`
  Compress files into `.bz2` format and remove the original files.
- `unzip_gz`
  Uncompress `.gz` files and remove the original archives.
- `zip_gz`
  Compress files into `.gz` format and remove the original files.

Dependencies:

- os
- glob
- bz2
- gzip

3.2.1 unzip_bz2

Uncompresses `.bz2` files and removes the original compressed file.

Parameters:

- `filepath`: str
  File path pattern to compressed `.bz2` files, using '*' wildcard characters.

Returns:

- `None`
  This function does not return any value.

Raises:

- `FileNotFoundError`

Sample Code:

```
import mydarn

mydarn.Convert.unzip_bz2('file/*.fitacf.bz2')
```

3.2.2 zip_bz2

Compresses files into `.bz2` format and removes the original uncompressed files.

Parameters:

- `filepath`: str
  File path pattern to uncompressed files, using '*' wildcard characters.

Returns:

- `None`
  This function does not return any value.

Raises:

- `FileNotFoundError`

Sample Code:

```
import mydarn

mydarn.Convert.zip_bz2('file/*.fitacf')
```

3.2.3 unzip_gz

Uncompresses `.gz` files and removes the original compressed file.

Parameters:

- `filepath`: str
  File path pattern to compressed `.gz` files, using '*' wildcard characters.

Returns:

- `None`
  This function does not return any value.

Raises:

- `FileNotFoundError`

Sample Code:

```
import mydarn

mydarn.Convert.unzip_gz('file/*.fitacf.gz')
```

3.2.4 zip_gz

Compresses files into `.gz` format and removes the original uncompressed files.

Parameters:

- `filepath`: str
  File path pattern to uncompressed files, using '*' wildcard characters.

Returns:

- `None`
  This function does not return any value.

Raises:

- `FileNotFoundError`

Sample Code:

```
import mydarn

mydarn.Convert.zip_gz('file/*.fitacf')
```

3.3 Filter

Provides methods to filter SuperDARN data by a chosen parameter within a specified range.

Methods:

- `filter_fitacf_gates`
  Filter FITACF data by range gate limits.

Dependencies:

- NumPy
- copy

3.3.1 filter_fitacf_gates

Filters SuperDARN FITACF data by range gate.

Parameters:

- `data`: list of dict
  A list of SuperDARN FITACF records.
- `gate_min`: int
  Minimum range gate (0 to 74), inclusive.
- `gate_max`: int
  Maximum range gate (0 to 74), inclusive.

Returns:

- `filtered_data`: list of dict
  FITACF records that fall within the specified range gate limits.

Sample Code:

```python
import mydarn
import pydarn

fitacf_data = pydarn.SuperDARNRead().read_dmap('file/name.fitacf')
filtered_data = mydarn.Filter.filter_fitacf_gates(fitacf_data, 10, 25)
```

3.4 RST

Processes SuperDARN data using RST subprocesses. Requires a working local installation of the Radar Software Toolkit (RST) version 5.1 or later.

Methods:

- `rawacf_to_fitacf`
  Convert RAWACF files to FITACF using the RST `make_fit` subprocess.
- `fitacf_to_grid`
  Convert FITACF files to GRID using the RST `make_grid` subprocess.
- `fit_speck_removal`
  Remove isolated noise points from FITACF data using the RST `fit_speck_removal` subprocess.

Dependencies:

- os
- subprocess
- glob

3.4.1 rawacf_to_fitacf

Converts SuperDARN RAWACF files to FITACF files using the RST `make_fit` subprocess.

Parameters:

- `rawacf_filepath`: str
  File path pattern to RAWACF file(s) containing '*' wildcard characters.
- `fitacf_filepath`: str
  File path to the output FITACF file to be created.
- `version`: float
  The fitting algorithm version to use (2.5 or 3).
  Default: 3

Returns:

- `None`
  This function does not return any value.

Raises:

- `ValueError`
- `FileNotFoundError`
- `OSError`

Sample Code:

```
import mydarn

mydarn.RST.rawacf_to_fitacf('file/*.rawacf', 'file/', 3)
```

3.4.2 fitacf_to_grid

Converts SuperDARN FITACF files to GRID files using the RST `make_grid` subprocess.

Parameters:

- `fitacf_filepath`: str
  File path pattern to FITACF file(s) containing '*' wildcard characters.
- `grid_filepath`: str
  File path to the output GRID file to be created.

Returns:

- `None`
  This function does not return any value.

Raises:

- `ValueError`
- `FileNotFoundError`
- `OSError`

Sample Code:

```
import mydarn

mydarn.RST.fitacf_to_grid('file/*.fitacf', 'file/')
```

3.4.3 fit_speck_removal

Removes isolated noise points from SuperDARN FITACF data using the RST `fit_speck_removal` subprocess.

Parameters:

- `fitacf_filepath`: str
  File path pattern to FITACF file(s) containing '*' wildcard characters.
- `despeck_file`: str
  File path to the output despecked FITACF file to be created.

Returns:

- `pct_removed`: list of float
  Percentage of noise points removed from the data by the `fit_speck_removal` subprocess.

Raises:

- `ValueError`
- `FileNotFoundError`
- `OSError`

Sample Code:

```
import mydarn

pct_removed = mydarn.RST.fit_speck_removal('file/*.fitacf', 'file/')
```

4 Plotting

4.1 Fan

Generates fan plots for SuperDARN data.

Methods:

- `plot_fan_plots`
  Create a sequence of fan plots for a specified parameter over a given time range.

Dependencies:

- pyDARN
- matplotlib
- datetime

4.1.1 plot_fan_plots

Plots a sequence of fan plots for a specified parameter from `start_time` to `end_time`, generating plots at intervals defined by `interval`. Each plot includes data within the specified time `tolerance` around the scan time.

Parameters:

- `data`: list of dict
  A list of SuperDARN FITACF records.
- `parameter`: str
  Key name of the vector field parameter to plot. Supported quantities include:
  - Line of sight velocity (`'v'`)
  - Power (`'p_l'`)
  - Spectral width (`'w_l'`)
  - Elevation angle (`'elv'`)
- `fig_path`: str
  File path to save each generated plot. If `None`, plots are not saved.
  Default: `None`
- `start_time`: `datetime.datetime`
  Datetime object marking the start of the plotting interval.
- `end_time`: `datetime.datetime`
  Datetime object marking the end of the plotting interval.
- `tolerance`: int
  Time window (in seconds) around each scan during which data is included.
  Default: 60
- `interval`: int
  Number of minutes between each generated plot.
  Default: 60

- `groundscatter`: bool
  If True, ground scatter points are shown in grey.
  Default: False
- `coastline`: bool
  If True, coastlines are drawn on the plot.
  Default: True
- `grid`: bool
  If True, overlays the radar field-of-view grid.
  Default: True
- `boundary`: bool
  If True, draws the outline of the radar field of view.
  Default: True
- `lowlat`: int
  Minimum latitude threshold. Only data above this latitude is plotted.
  Default: 0

Returns:

- `None`
  This function does not return any value.

Raises:

- `TypeError`
- `ValueError`

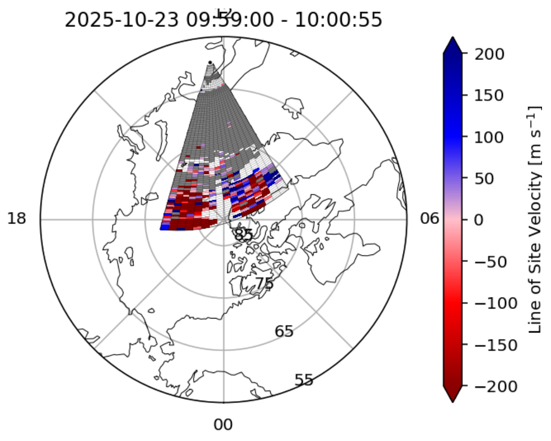
Sample Code:

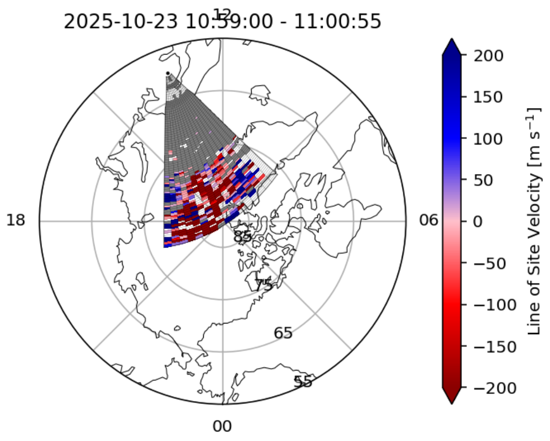
```
import mydarn
import pydarn
from datetime import datetime

start = datetime(2025, 10, 23, 10, 00)
end = datetime(2025, 10, 23, 11, 00)

fitacf_data = pydarn.SuperDARNRead().read_dmap('file/name.fitacf')
mydarn.Fan.plot_fan_plots(fitacf_data, 'v', start_time = start, end_time = end,
                          interval = 60, tolerance = 60, groundscatter = True,
                          lowlat = 55, fig_path = 'file/')
```

Output:


2025-10-23 09:59:00 - 10:00:55
12
18
06
00
85
75
65
55
200
150
100
50
0
−50
−100
−150
−200
Line of Site Velocity [m s−1]


2025-10-23 10:59:00 - 11:00:55
12
18
06
00
85
75
65
55
200
150
100
50
0
−50
−100
−150
−200
Line of Site Velocity [m s−1]

4.2 Histogram

Generates one-dimensional and two-dimensional histogram plots for SuperDARN data.

Methods:

- `plot_vector_hist`
  Plot histograms of vector field quantities.
- `plot_scalar_hist`
  Plot histograms of scalar field quantities.
- `plot_2d_hist`
  Plot 2D histograms of vector field quantities versus range gate.
- `plot_freq_scan_hist`
  Plot histograms of ionospheric scatter occurrence as a function of transmitted frequency band.
- `plot_freq_scan_2d_hist`
  Plot 2D histograms of ionospheric scatter occurrence as a function of frequency band and time.

Dependencies:

- pyDARN
- NumPy
- matplotlib
- datetime

4.2.1 plot_vector_hist

Plots a histogram of the specified vector field quantity.

Parameters:

- `parameter`: str
  Key name of the vector field parameter to plot. Supported quantities include:
  - Line of sight velocity (`'v'`)
  - Power (`'p_l'`)
  - Spectral width (`'w_l'`)
  - Elevation angle (`'elv'`)
  - Lag-zero power (`'pwr0'`)
  - Phase offset (`'phi0'`)
- `data`: list of dict
  A list of SuperDARN FITACF or RAWACF records.
- `groundscatter`: bool
  If True, ground scatter values for the specified parameter are plotted in a separate histogram and statistics are returned separately.

Default: False

- `normalize`: bool
  If True, the histogram is plotted as a probability density (area normalized to 1).
  Default: True

- `num_bins`: int or str
  If an int, specifies the number of equal-width bins in the histogram. If a str, specifies the method used to determine the number of bins.
  Default: `'auto'`

- `boundary`: str or (float, float)
  The lower and upper range of the bins; values outside this range are ignored. If `None`, uses `(min(vals), max(vals))`. If `'auto'`, the lowest 0.5% and highest 0.5% of the data values are excluded.
  Default: None

- `num_bins_gs`: int or str
  As `num_bins`, but applied to the ground scatter histogram.
  Default: `'auto'`

- `boundary_gs`: str or (float, float)
  As `boundary`, but applied to the ground scatter histogram.
  Default: None

- `beam_num`: int or str
  If an int, selects data from a single beam number (0 to 15) to include. If a str, data from all beam numbers are included.
  Default: `'all'`

- `gate_num`: int or str
  If an int, selects data from a single gate number (0 to 74) to include. If a str, data from all gate numbers are included.
  Default: `'all'`

- `show_txt`: bool
  If True, the number of bins, standard deviation, mean, median, and mode are annotated on the histogram.
  Default: True

Returns:

- `std`: float
  Standard deviation of the data in the histogram. Includes ground scatter values if `groundscatter` is False.

- `mean`: float
  Mean of the data in the histogram. Includes ground scatter values if `groundscatter` is False.

- `median`: float
  Median of the data in the histogram. Includes ground scatter values if `groundscatter` is False.

- `mode`: float
  Mode of the data in the histogram. Includes ground scatter values if `groundscatter` is False.

- `num_pts`: int
  Number of points in the dataset for the specified parameter. Includes ground scatter values if `groundscatter` is False.

- `std_gs`: float

Standard deviation of the ground scatter data in the histogram. Only returned if `groundscatter` is True.

- `mean_gs`: float
  Mean of the ground scatter data in the histogram. Only returned if `groundscatter` is True.
- `median_gs`: float
  Median of the ground scatter data in the histogram. Only returned if `groundscatter` is True.
- `mode_gs`: float
  Mode of the ground scatter data in the histogram. Only returned if `groundscatter` is True.
- `num_pts_gs`: int
  Number of points in the ground scatter dataset for the specified parameter. Only returned if `groundscatter` is True.
- `pct_gs`: float
  Percentage of data flagged as ground scatter by the `gflg` parameter. Only returned if `groundscatter` is True.

Raises:

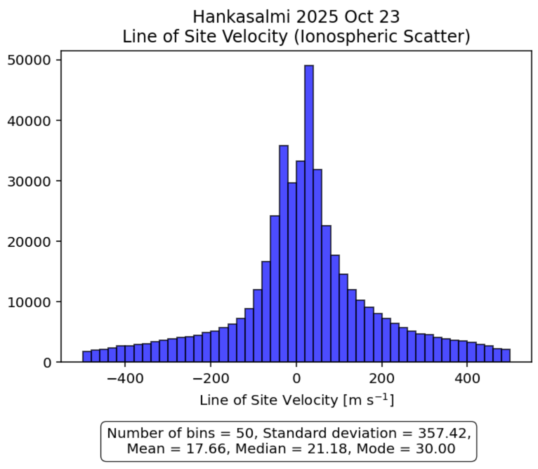
- `KeyError`
- `ValueError`

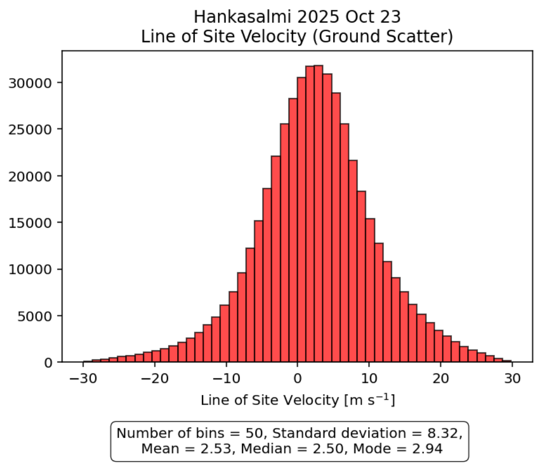
Sample Code:

```python
import mydarn
import pydarn
import matplotlib.pyplot as plt

fitacf_data = pydarn.SuperDARNRead().read_dmap('file/name.fitacf')
mydarn.Histogram.plot_vector_hist('v', fitacf_data, groundscatter = True,
                                  boundary = (-500, 500), normalize = False,
                                  num_bins = 50, num_bins_gs = 50)
plt.show()
```

Output:





```
{'std': 357.4194641113281,
 'mean': 17.66352653503418,
 'median': 21.17535972595215,
 'mode': 30.0,
 'num_pts': 524449,
 'std_gs': 8.318714141845703,
 'mean_gs': 2.5274786949157715,
 'median_gs': 2.5042030811309814,
 'mode_gs': 2.943920135498047,
 'num_pts_gs': 475551,
 'pct_gs': 47.5551}
```

4.2.2 plot_scalar_hist

Plots a histogram of the specified scalar field quantity.

Parameters:

- `parameter`: str
  Key name of the scalar field parameter to plot. Supported quantities include:
  - Sky noise (`'noise.sky'`)
  - Transmitted frequency (`'tfreq'`)
  - Lag to first range (`'lagfr'`)
  - Sample separation (`'smsep'`)
  - Number of pulse sequences transmitted (`'nave'`)
- `data`: list of dict
  A list of SuperDARN FITACF or RAWACF records.
- `normalize`: bool
  If True, the histogram is plotted as a probability density (area normalized to 1).
  Default: True
- `num_bins`: int or str
  If an int, specifies the number of equal-width bins in the histogram. If a str, specifies the method used to determine the number of bins.
  Default: `'auto'`
- `boundary`: str or (float, float)
  The lower and upper range of the bins; values outside this range are ignored. If `None`, uses `(min(vals), max(vals))`. If `'auto'`, the lowest 0.5% and highest 0.5% of the data values are excluded.
  Default: None
- `beam_num`: int or str
  If an int, selects data from a single beam number (0 to 15) to include. If a str, data from all beam numbers are included.
  Default: `'all'`
- `show_txt`: bool
  If True, the number of bins, standard deviation, mean, median, and mode are annotated on the histogram.
  Default: True

Returns:

- `std`: float
  Standard deviation of the data in the histogram.
- `mean`: float
  Mean of the data in the histogram.
- `median`: float
  Median of the data in the histogram.
- `mode`: float
  Mode of the data in the histogram.
- `num_pts`: int
  Number of points in the dataset for the specified parameter.

Raises:

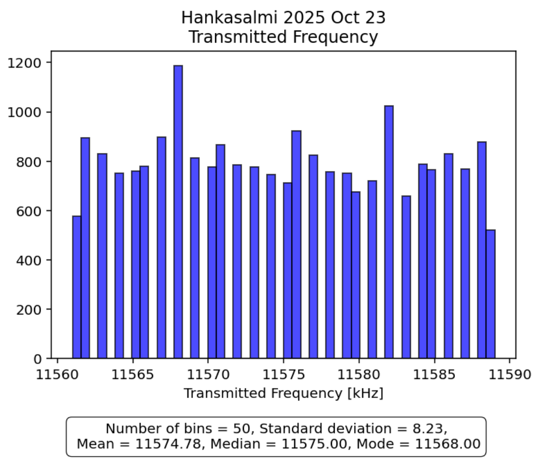
- `KeyError`
- `ValueError`

Sample Code:

```
import mydarn
import pydarn
import matplotlib.pyplot as plt

fitacf_data = pydarn.SuperDARNRead().read_dmap('file/name.fitacf')
mydarn.Histogram.plot_scalar_hist('tfreq', fitacf_data, normalize = False,
                                  num_bins = 50)
plt.show()
```

Output:



```
{'std': 8.234195500608017,
 'mean': 11574.779079861111,
 'median': 11575.0,
 'mode': 11568.0,
 'num_pts': 23040}
```

4.2.3 plot_2d_hist

Plots a 2D histogram of the specified vector field quantity along the y-axis and range gate number along the x-axis.

Parameters:

- `parameter`: str
  Key name of the vector field parameter to plot. Supported quantities include:
  - Line of sight velocity (`'v'`)
  - Power (`'p_l'`)
  - Spectral width (`'w_l'`)
  - Elevation angle (`'elv'`)
  - Lag-zero power (`'pwr0'`)
  - Phase offset (`'phi0'`)
- `data`: list of dict
  A list of SuperDARN FITACF or RAWACF records.
- `groundscatter`: bool
  If True, ground scatter values for the specified parameter are plotted in a separate 2D histogram, and corresponding statistics are returned separately.
  Default: False
- `normalize`: bool
  If True, the histogram is plotted as a probability density (area normalized to 1).
  Default: True
- `num_bins`: int or str
  If an int, specifies the number of equal-width bins used along the y-axis. If a str, specifies the method used to determine the number of bins.
  Default: `'auto'`
- `boundary`: str or (float, float)
  The lower and upper bounds of the data range along the y-axis; values outside this range are ignored. If `None`, uses `(min(vals), max(vals))`. If `'auto'`, the lowest 0.5% and highest 0.5% of values are excluded.
  Default: None
- `num_bins_gs`: int or str
  As `num_bins`, but applied to the ground scatter histogram.
  Default: `'auto'`
- `boundary_gs`: str or (float, float)
  As `boundary`, but applied to the ground scatter histogram.
  Default: None
- `beam_num`: int or str
  If an int, selects data from a single beam number (0 to 15) to include. If a str, data from all beam numbers are included.
  Default: `'all'`
- `cmap`: str
  Matplotlib colormap used when rendering the histogram.
  Default: `'plasma'`

Returns:

- `num_bins`: int
  Number of bins used to generate the histogram along each axis. Includes ground scatter values if `groundscatter` is False.
- `num_pts`: int
  Number of points in the dataset for the specified parameter. Includes ground scatter values if `groundscatter` is False.
- `num_bins_gs`: int
  Number of bins used for the ground scatter histogram along each axis. Only returned if `groundscatter` is True.
- `num_pts_gs`: int
  Number of points in the ground scatter dataset for the specified parameter. Only returned if `groundscatter` is True.

Raises:

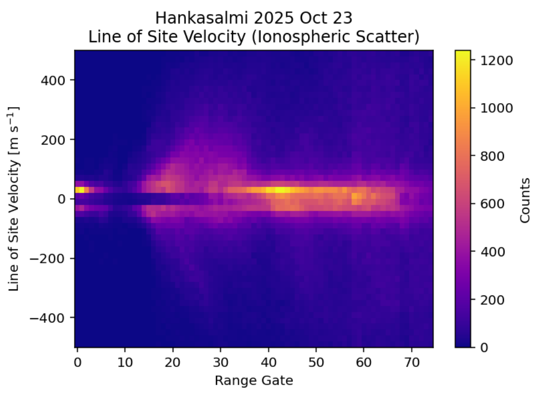
- `KeyError`
- `ValueError`

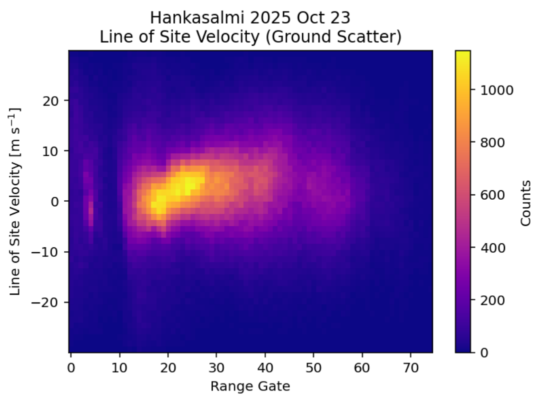
Sample Code:

```
import mydarn
import pydarn
import matplotlib.pyplot as plt

fitacf_data = pydarn.SuperDARNRead().read_dmap('file/name.fitacf')
mydarn.Histogram.plot_2d_hist('v', fitacf_data, groundscatter = True,
                              normalize = False, boundary = (-500, 500),
                              num_bins = 50, num_bins_gs = 50)
plt.show()
```

Output:





```
{'num_bins': [75, 50],
 'num_bins_gs': [75, 50],
 'num_pts': 524449,
 'num_pts_gs': 475551}
```

4.2.4 plot_freq_scan_hist

Plots a histogram of the number of ionospheric scatter points as a function of the transmitted frequency band. This function is intended for plotting data from frequency sweeps and uses data from channel B only.

Parameters:

- `data`: list of dict
  A list of SuperDARN FITACF records.

- `freq_bands`: dict
  Dictionary specifying the frequency bands. Keys are band identifiers, and values are two-element lists giving the frequency range (in kHz) for each band.

- `normalize`: bool
  If True, the histogram is plotted as a probability density (area normalized to 1).
  Default: True

- `boundary`: str or (float, float)
  The lower and upper frequency limits used for binning; values outside this range are ignored. If `None`, uses the minimum and maximum frequencies defined in `freq_bands`.
  Default: None

- `omit_bands`: int or list of int
  Frequency band numbers to omit from the histogram and from the ordering in the returned result.
  Default: None

- `show_txt`: bool
  If True, the optimal frequency band (the band with the highest number of ionospheric scatter points) is annotated on the histogram.
  Default: True

Returns:

- `ordered_bands`: list of dict
  Frequency bands ordered from the highest to the lowest number of ionospheric scatter points. Includes the band indentifier, frequency range for each band, and number of ionospheric scatter points within each band.

Raises:

- `TypeError`

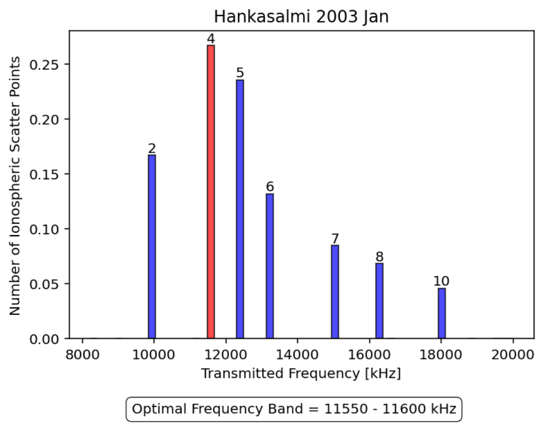
Sample Code:

```
import mydarn
import pydarn
import matplotlib.pyplot as plt

freq_bands = {"0": [8305, 8335],
              "1": [8965, 9040],
              "2": [9900, 9985],
              "3": [11075, 11275],
              "4": [11550, 11600],
              "5": [12370, 12415],
              "6": [13200, 13260],
              "7": [15010, 15080],
              "8": [16210, 16360],
              "9": [16555, 16615],
              "10": [17970, 18050],
              "11": [18850, 18865],
              "12": [19415, 19680],
              "13": [19705, 19755],
              "14": [19800, 19990]}

freq_sweep_data = pydarn.SuperDARNRead().read_dmap('file/name.fitacf')
mydarn.Histogram.plot_freq_scan_hist(freq_sweep_data, freq_bands)
plt.show()
```

Output:



```
[{'Band': 4, 'Freq Range': [11550, 11600], 'Count': 12435},
 {'Band': 5, 'Freq Range': [12370, 12415], 'Count': 10971},
 {'Band': 2, 'Freq Range': [9900, 9985], 'Count': 7781},
 {'Band': 6, 'Freq Range': [13200, 13260], 'Count': 6150},
 {'Band': 7, 'Freq Range': [15010, 15080], 'Count': 3946},
 {'Band': 8, 'Freq Range': [16210, 16360], 'Count': 3181},
 {'Band': 10, 'Freq Range': [17970, 18050], 'Count': 2135},
 {'Band': 14, 'Freq Range': [19800, 19990], 'Count': 0},
 {'Band': 13, 'Freq Range': [19705, 19755], 'Count': 0},
 {'Band': 12, 'Freq Range': [19415, 19680], 'Count': 0},
 {'Band': 11, 'Freq Range': [18850, 18865], 'Count': 0},
 {'Band': 9, 'Freq Range': [16555, 16615], 'Count': 0},
 {'Band': 3, 'Freq Range': [11075, 11275], 'Count': 0},
 {'Band': 1, 'Freq Range': [8965, 9040], 'Count': 0},
 {'Band': 0, 'Freq Range': [8305, 8335], 'Count': 0}]
```

4.2.5 plot_freq_scan_2d_hist

Plots a 2D histogram of the number of ionospheric scatter points with frequency band along the y-axis and time (year, month, day, or hour) along the x-axis. This function is intended for plotting data from frequency sweeps and uses data from channel B only.

Parameters:

- `data`: list of dict
  A list of SuperDARN FITACF records.
- `freq_bands`: dict
  Dictionary specifying the frequency bands. Keys are band identifiers, and values are two-element lists giving the frequency range (in kHz) for each band.
- `date_bin`: str
  Time scale used to bin the data along the x-axis. Supported options include:
  - `'year'`
  - `'month'`
  - `'day'`
  - `'hour'`
- `normalize`: bool
  If True, each column of the 2D histogram is normalized such that the column sums to 1 (column-wise probability density).
  Default: True
- `boundary`: str or (float, float)
  The lower and upper frequency limits used for binning. Values outside this range are ignored. If `None`, uses the minimum and maximum frequencies defined in `freq_bands`.
  Default: None
- `omit_bands`: int or list of int
  Frequency band numbers to omit from the histogram and from the ordering in the returned result.
  Default: None
- `cmap`: str
  Matplotlib colormap used to render the 2D histogram.
  Default: `'plasma'`
- `show_txt`: bool
  If True, the optimal frequency band (the band with the highest number of ionospheric scatter points) is annotated on the histogram.
  Default: True

Returns:

- `ordered_bands`: list of dict
  Frequency bands ordered from the highest to the lowest number of ionospheric scatter points. Includes the band indentifier, frequency range for each band, and number of ionospheric scatter points within each band.

Raises:

- ValueError
- TypeError

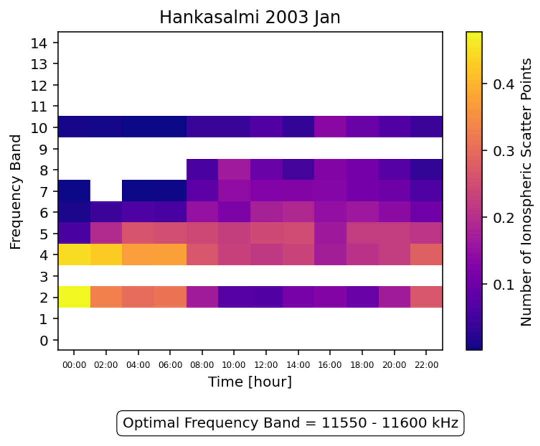
Sample Code:

```
import mydarn
import pydarn
import matplotlib.pyplot as plt

freq_bands = {"0": [8305, 8335],
              "1": [8965, 9040],
              "2": [9900, 9985],
              "3": [11075, 11275],
              "4": [11550, 11600],
              "5": [12370, 12415],
              "6": [13200, 13260],
              "7": [15010, 15080],
              "8": [16210, 16360],
              "9": [16555, 16615],
              "10": [17970, 18050],
              "11": [18850, 18865],
              "12": [19415, 19680],
              "13": [19705, 19755],
              "14": [19800, 19990]}

freq_sweep_data = pydarn.SuperDARNRead().read_dmap('file/name.fitacf')
mydarn.Histogram.plot_freq_scan_2d_hist(freq_sweep_data, freq_bands,
                                        date_bin = 'hour')
plt.show()
```

Output:



```
[{'Band': 4, 'Freq Range': [11550, 11600], 'Count': 12435},
 {'Band': 5, 'Freq Range': [12370, 12415], 'Count': 10971},
 {'Band': 2, 'Freq Range': [9900, 9985], 'Count': 7781},
 {'Band': 6, 'Freq Range': [13200, 13260], 'Count': 6150},
 {'Band': 7, 'Freq Range': [15010, 15080], 'Count': 3946},
 {'Band': 8, 'Freq Range': [16210, 16360], 'Count': 3181},
 {'Band': 10, 'Freq Range': [17970, 18050], 'Count': 2135},
 {'Band': 14, 'Freq Range': [19800, 19990], 'Count': 0},
 {'Band': 13, 'Freq Range': [19705, 19755], 'Count': 0},
 {'Band': 12, 'Freq Range': [19415, 19680], 'Count': 0},
 {'Band': 11, 'Freq Range': [18850, 18865], 'Count': 0},
 {'Band': 9, 'Freq Range': [16555, 16615], 'Count': 0},
 {'Band': 3, 'Freq Range': [11075, 11275], 'Count': 0},
 {'Band': 1, 'Freq Range': [8965, 9040], 'Count': 0},
 {'Band': 0, 'Freq Range': [8305, 8335], 'Count': 0}]
```

4.3 RTP

Generates time series plots of vector and scalar fiel parameters for SuperDARN data.

Methods:

- `plot_vector_time_series`
  Plot the time series of a selected vector field parameter for a given range gate and beam number.
- `plot_scalar_time_series`
  Plot the time series of a selected scalar field parameter for a given beam number.

Dependencies:

- pyDARN
- NumPy
- matplotlib
- datetime

4.3.1 plot_vector_time_series

Plots a time series of the specified vector field parameter for a given range gate and beam number.

Future Work:

- Plot the time series of the specified vector field parameter across all range gates for a given beam number, or across all beam numbers for a given range gate.
- Plot the time series of the specified vector field parameter averaged across a specified range gate or beam number.

Parameters:

- `parameter`: str
  Key name of the vector field parameter to plot. Supported quantities include:
  - Line of sight velocity (`'v'`)
  - Power (`'p_l'`)
  - Spectral width (`'w_l'`)
  - Elevation angle (`'elv'`)
  - Lag-zero power (`'pwr0'`)
  - Phase offset (`'phi0'`)
- `data`: list of dict
  A list of SuperDARN FITACF or RAWACF records.
- `beam_num`: int
  Beam number (0 to 15) of the data to plot.
- `gate_num`: int
  Range gate number (0 to 74) of the data to plot.

- `groundscatter`: bool
  If True, ground scatter values for the specified parameter are plotted separately from ionospheric scatter.
  Default: False

Returns:

- `None`
  This function does not return any value.

Raises:

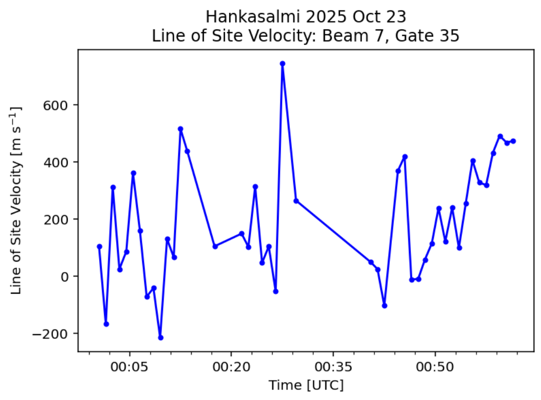
- `KeyError`
- `ValueError`

Sample Code:

```
import mydarn
import pydarn
import matplotlib.pyplot as plt

fitacf_data = pydarn.SuperDARNRead().read_dmap('file/name.fitacf')
mydarn.RTP.plot_vector_time_series('v', fitacf_data,
                                   beam_num = 7, gate_num = 35)
plt.show()
```

Output:



4.3.2 plot_scalar_time_series

Plots a time series of the specified scalar field parameter for a given beam number.

Future Work:

- Plot the time series of the specified scalar field parameter across all beam numbers.
- Plot the time series of the specified scalar field parameter averaged across a specified beam number.

Parameters:

- `parameter`: str
  Key name of the scalar field parameter to plot. Supported quantities include:
  - Sky noise (`'noise.sky'`)
  - Transmitted frequency (`'tfreq'`)
  - Lag to first range (`'lagfr'`)
  - Sample separation (`'smsep'`)
  - Number of pulse sequences transmitted (`'nave'`)
- `data`: list of dict
  A list of SuperDARN FITACF or RAWACF records.
- `beam_num`: int
  Beam number (0 to 15) of the data to plot.

Returns:

- `None`
  This function does not return any value.

Raises:

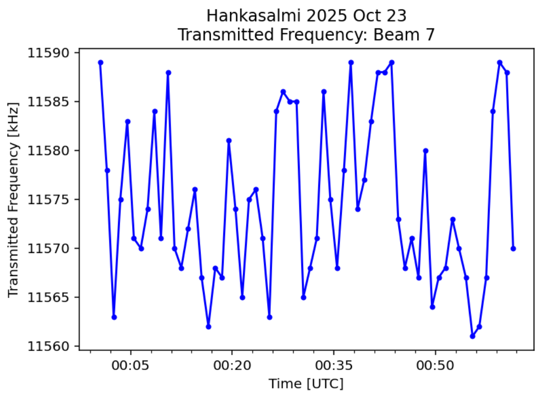
- `KeyError`
- `ValueError`

Sample Code:

```
import mydarn
import pydarn
import matplotlib.pyplot as plt

fitacf_data = pydarn.SuperDARNRead().read_dmap('file/name.fitacf')
mydarn.RTP.plot_vector_time_series('tfreq', fitacf_data, beam_num = 7)
plt.show()
```

Output:



4.4 Scatter

Generates scatter plots for SuperDARN data.

Methods:

- `plot_range_scatter`
  Plot a scatter plot of a selected vector field parameter as a function of range gate number.

Dependencies:

- pyDARN
- NumPy
- matplotlib
- datetime

4.4.1 plot_range_scatter

Plots a scatter plot of the specified vector field parameter on the y-axis and range gate number on the x-axis.

Parameters:

- `parameter`: str
  Key name of the vector field parameter to plot. Supported quantities include:
  - Line of sight velocity (`'v'`)
  - Power (`'p_l'`)
  - Spectral width (`'w_l'`)
  - Elevation angle (`'elv'`)
  - Lag-zero power (`'pwr0'`)
  - Phase offset (`'phi0'`)
- `data`: list of dict
  A list of SuperDARN FITACF or RAWACF records.
- `groundscatter`: bool
  If True, ground scatter values for the specified parameter are plotted in a separate scatter plot.
  Default: False
- `beam_num`: int or str
  If an int, selects data from a single beam number (0 to 15). If a str, data from all beam numbers are included.
  Default: `'all'`

Returns:

- `None`
  This function does not return any value.

Raises:

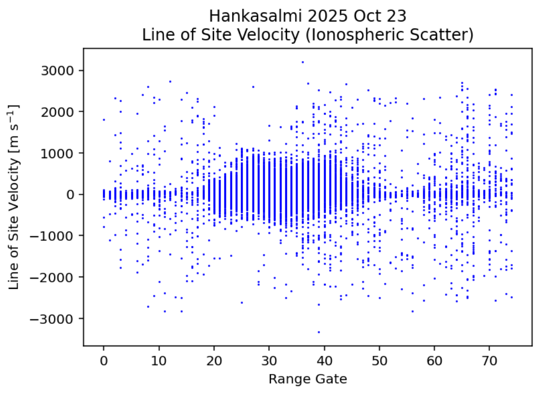
- KeyError
- ValueError

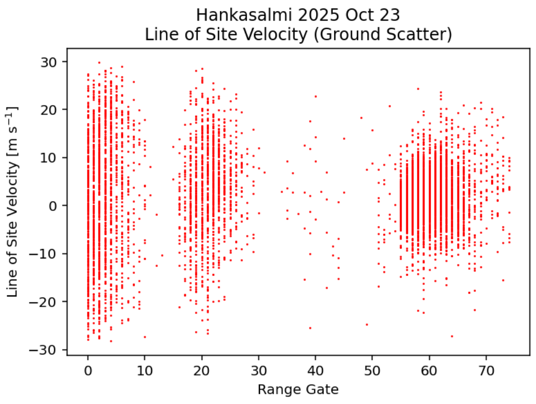
Sample Code:

```
import mydarn
import pydarn
import matplotlib.pyplot as plt

fitacf_data = pydarn.SuperDARNRead().read_dmap('file/name.fitacf')
mydarn.Scatter.plot_range_scatter('v', fitacf_data, groundscatter = True)
plt.show()
```

Output:





References

[1] SuperDARN Data Visualization Working Group, C. J. Martin, R. A. Rohel, D. D. Billett, P. Pitzer, D. Galeshuck, B. S. R. Kunduri, K. Khanal, A. Hiyadutuje, M. Chakraborty, S.and Detwiller, and M. T. Schmidt. SuperDARN/pyDARN: pyDARN v4.1 (v4.1), 2024.

[2] SuperDARN Data Visualization Working Group, R. A. Rohel, E. C. Bland, C. J. Martin, A. Chartier, K. J. Krieger, D. D. Billett, A. G. Burrell, M. H. Detwiller, K. Kotyk, M. T. Schmidt, and X. Shi. SuperDARN/pyDARNio: pyDARNio v1.2.1 (v1.2.1), 2023.

[3] SuperDARN Data Analysis Working Group, E. G. Thomas, A. G. Burrell, P. V. Ponomarenko, E. C. Bland, K. T. Sterne, R. A. Rohel, S. G. Shepherd, D. D. Billett, M. T. Schmidt, and M. T. Walach. SuperDARN Radar Software Toolkit (RST) 5.1 (v5.1), 2025.