windrose Documentation

Lionel Roubeyrie & Sebastien Celles

May 06, 2022

Contents:

CHAPTER 1

Install

1.1 Requirements

- matplotlib https://matplotlib.org/
- numpy https://numpy.org/
- and naturally python https://www.python.org/ :-P

Option libraries:

- Pandas https://pandas.pydata.org/ (to feed plot functions easily)
- SciPy https://scipy.org/ (to fit data with Weibull distribution)
- ffmpeg https://www.ffmpeg.org/ (to output video)
- click https://click.palletsprojects.com/ (for command line interface tools)

1.2 Install latest release version via pip

A package is available and can be downloaded from PyPi and installed using:

```
$ pip install windrose
```

1.3 Install latest development version

```
$ pip install git+https://github.com/python-windrose/windrose
```

or

```
$ git clone https://github.com/python-windrose/windrose
$ python setup.py install
```

CHAPTER 2

Notebook example

An IPython (Jupyter) notebook showing this package usage is available at:

- https://nbviewer.org/github/python-windrose/windrose/blob/master/windrose_sample_random.ipynb

CHAPTER 3

Script example

This example use randoms values for wind speed and direction(ws and wd variables). In situation, these variables are loaded with reals values (1-D array), from a database or directly from a text file (see the "load" facility from the matplotlib.pylab interface for that).

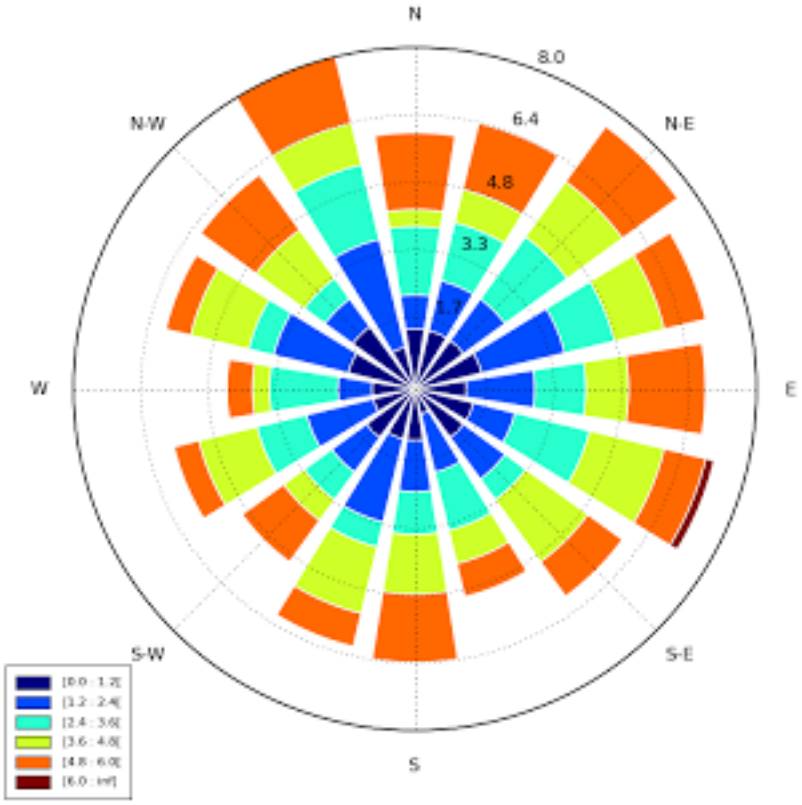
```
from windrose import WindroseAxes
from matplotlib import pyplot as plt
import matplotlib.cm as cm
import numpy as np

# Create wind speed and direction variables

ws = np.random.random(500) * 6
wd = np.random.random(500) * 360
```

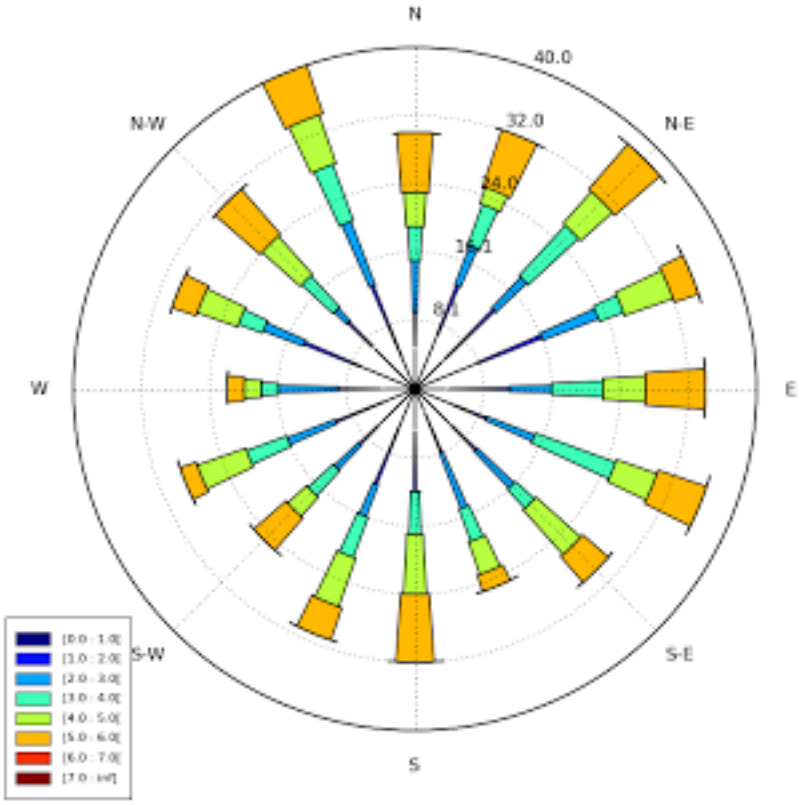
3.1 A stacked histogram with normed (displayed in percent) results

```
ax = WindroseAxes.from_ax()
ax.bar(wd, ws, normed=True, opening=0.8, edgecolor='white')
ax.set_legend()
```

3.2 Another stacked histogram representation, not normed, with bins limits

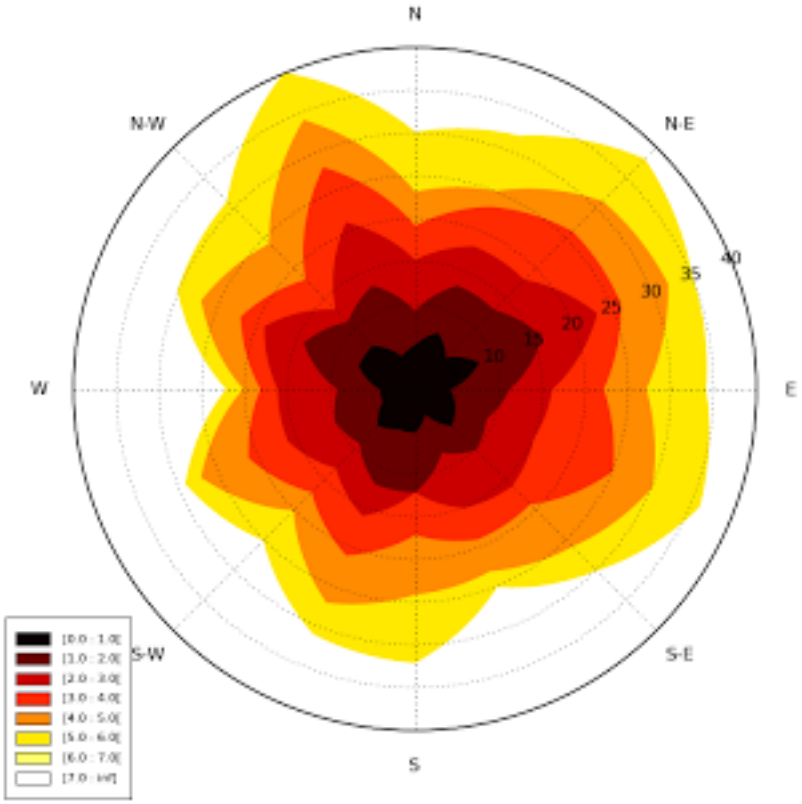
```
ax = WindroseAxes.from_ax()
ax.box(wd, ws, bins=np.arange(0, 8, 1))
ax.set_legend()
```


N
N-E
E
S-E
S
S-W
W
N-W
8.0
6.4
4.8
3.3
1.7
[0.0 - 1.2[
[1.2 - 2.4[
[2.4 - 3.6[
[3.6 - 4.8[
[4.8 - 6.0[
[6.0 - inf]


N
N-E
E
S-E
S
S-W
W
N-W
40.0
32.0
24.0
16.1
8.1
[0.0 - 1.0[
[1.0 - 2.0[
[2.0 - 3.0[
[3.0 - 4.0[
[4.0 - 5.0[
[5.0 - 6.0[
[6.0 - 7.0[
[7.0 - inf]

3.3 A windrose in filled representation, with a controlled colormap

```
ax = WindroseAxes.from_ax()
ax.contourf(wd, ws, bins=np.arange(0, 8, 1), cmap=cm.hot)
ax.set_legend()
```



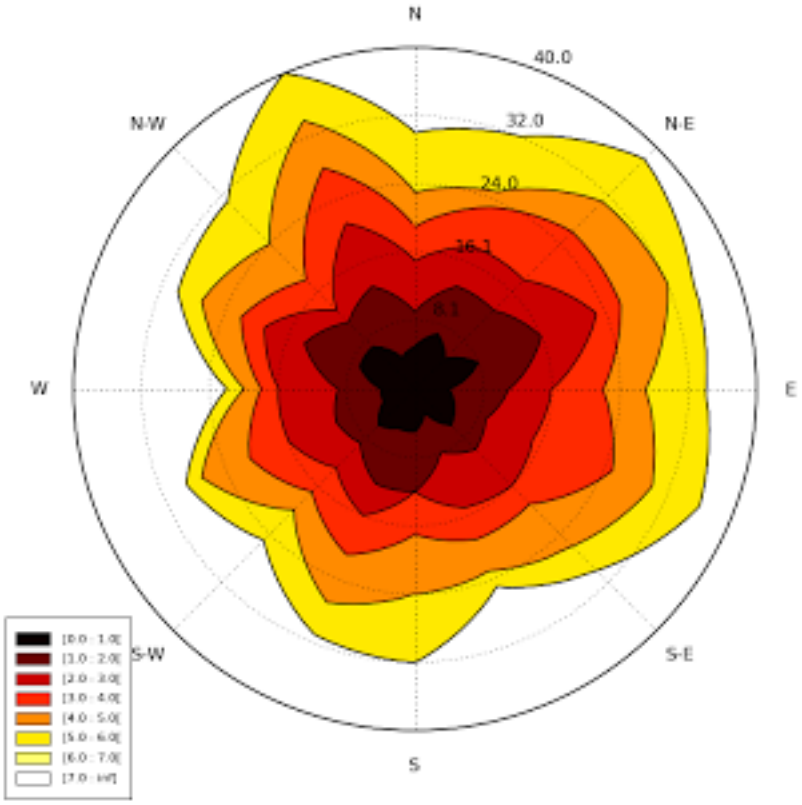
3.4 Same as above, but with contours over each filled region...

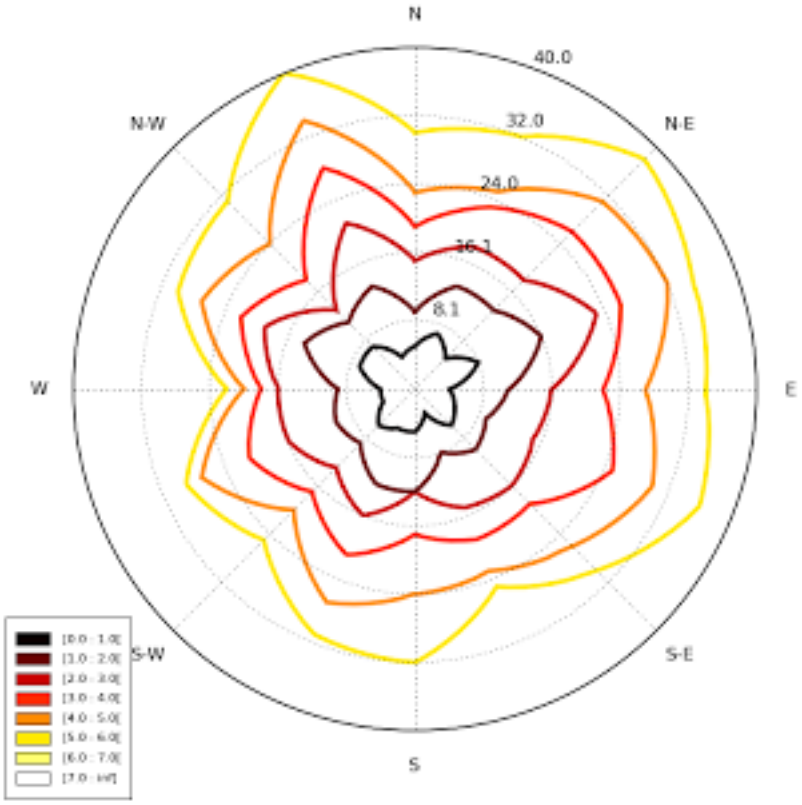
```
ax = WindroseAxes.from_ax()
ax.contourf(wd, ws, bins=np.arange(0, 8, 1), cmap=cm.hot)
ax.contour(wd, ws, bins=np.arange(0, 8, 1), colors='black')
ax.set_legend()
```

3.5 ...or without filled regions

```
ax = WindroseAxes.from_ax()
ax.contour(wd, ws, bins=np.arange(0, 8, 1), cmap=cm.hot, lw=3)
ax.set_legend()
```

After that, you can have a look at the computed values used to plot the windrose with the `ax._info` dictionary :


N
N-E
E
S-E
S
S-W
W
N-W
40.0
32.0
24.0
16.1
8.1
[0.0 - 1.0[
[1.0 - 2.0[
[2.0 - 3.0[
[3.0 - 4.0[
[4.0 - 5.0[
[5.0 - 6.0[
[6.0 - 7.0[
[7.0 - inf]


N
N-E
E
S-E
S
S-W
W
N-W
40.0
32.0
24.0
16.1
8.1
[0.0 - 1.0[
[1.0 - 2.0[
[2.0 - 3.0[
[3.0 - 4.0[
[4.0 - 5.0[
[5.0 - 6.0[
[6.0 - 7.0[
[7.0 - inf]

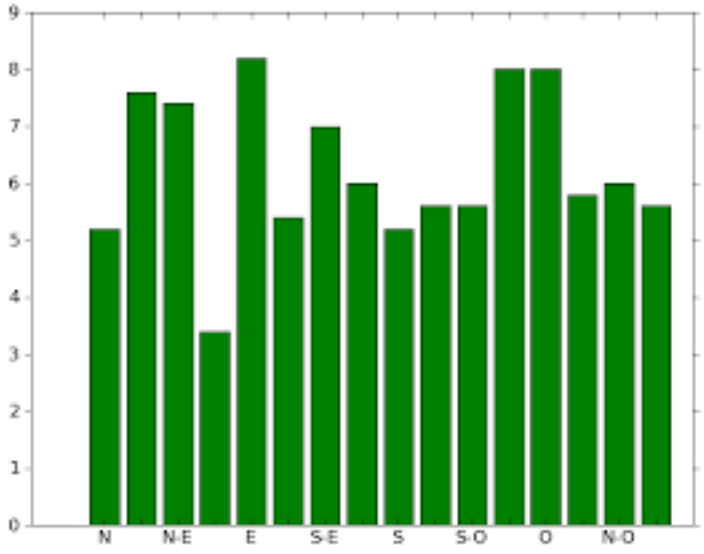
- `ax._info['bins']` : list of bins (limits) used for wind speeds. If not set in the call, bins will be set to 6 parts between wind speed min and max.
- `ax._info['dir']` : list of directions "boundaries" used to compute the distribution by wind direction sector. This can be set by the nsector parameter (see below).
- `ax._info['table']` : the resulting table of the computation. It's a 2D histogram, where each line represents a wind speed class, and each column represents a wind direction class.

So, to know the frequency of each wind direction, for all wind speeds, do:

```
ax.bar(wd, ws, normed=True, nsector=16)
table = ax._info['table']
wd_freq = np.sum(table, axis=0)
```

and to have a graphical representation of this result :

```
direction = ax._info['dir']
wd_freq = np.sum(table, axis=0)
plt.bar(np.arange(16), wd_freq, align='center')
xlabels = ('N','','N-E','','E','','S-E','','S','','S-O','','O','','N-O','')
xticks=arange(16)
gca().set_xticks(xticks)
draw()
gca().set_xticklabels(xlabels)
draw()
```



In addition of all the standard pyplot parameters, you can pass special parameters to control the windrose production. For the stacked histogram windrose, calling help(ax.bar) will give : `bar(self, direction, var, **kwargs)` method of `windrose.WindroseAxes` instance Plot a windrose in bar mode. For each var bins and for each sector, a colored bar will be draw on the axes.

Mandatory:

- `direction` : 1D array - directions the wind blows from, North centred
- `var` : 1D array - values of the variable to compute. Typically the wind speeds

Optional:

- `nsector` : integer - number of sectors used to compute the windrose table. If not set, nsectors=16, then each sector will be 360/16=22.5°, and the resulting computed table will be aligned with the cardinals points.
- `bins` : 1D array or integer - number of bins, or a sequence of bins variable. If not set, bins=6 between min(var) and max(var).
- `blowto` : bool. If True, the windrose will be pi rotated, to show where the wind blow to (useful for pollutant rose).
- `colors` : string or tuple - one string color (`'k'` or `'black'`), in this case all bins will be plotted in this color; a tuple of matplotlib color args (string, float, rgb, etc), different levels will be plotted in different colors in the order specified.
- `cmap` : a cm Colormap instance from `matplotlib.cm`. - if `cmap == None` and `colors == None`, a default Colormap is used.
- `edgecolor` : string - The string color each edge bar will be plotted. Default : no edgecolor
- `opening` : float - between 0.0 and 1.0, to control the space between each sector (1.0 for no space)
- `mean_values` : Bool - specify wind speed statistics with direction=specific mean wind speeds. If this flag is specified, var is expected to be an array of mean wind speeds corresponding to each entry in `direction`. These are used to generate a distribution of wind speeds assuming the distribution is Weibull with shape factor = 2.
- `weibull_factors` : Bool - specify wind speed statistics with direction=specific weibull scale and shape factors. If this flag is specified, var is expected to be of the form [[7,2], ..., [7.5,1.9]] where var[i][0] is the weibull scale factor and var[i][1] is the shape factor

3.6 probability density function (pdf) and fitting Weibull distribution

A probability density function can be plot using:

```
from windrose import WindAxes
ax = WindAxes.from_ax()
bins = np.arange(0, 6 + 1, 0.5)
bins = bins[1:]
ax, params = ax.pdf(ws, bins=bins)
```

Optimal parameters of Weibull distribution can be displayed using

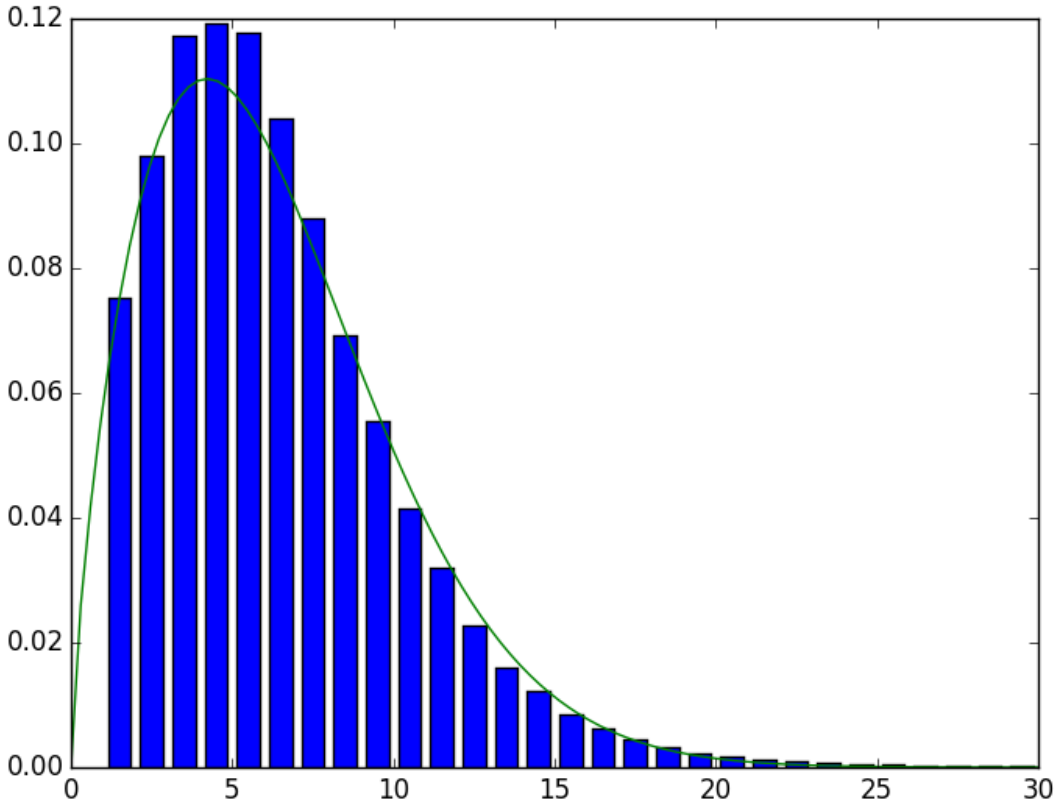
```
print(params)
(1, 1.7042156870194352, 0, 7.0907180300605459)
```

3.7 Overlay of a map

This example illustrate how to set an windrose axe on top of any other axes. Specifically, overlaying a map is often useful. It rely on matplotlib toolbox inset_axes utilities.

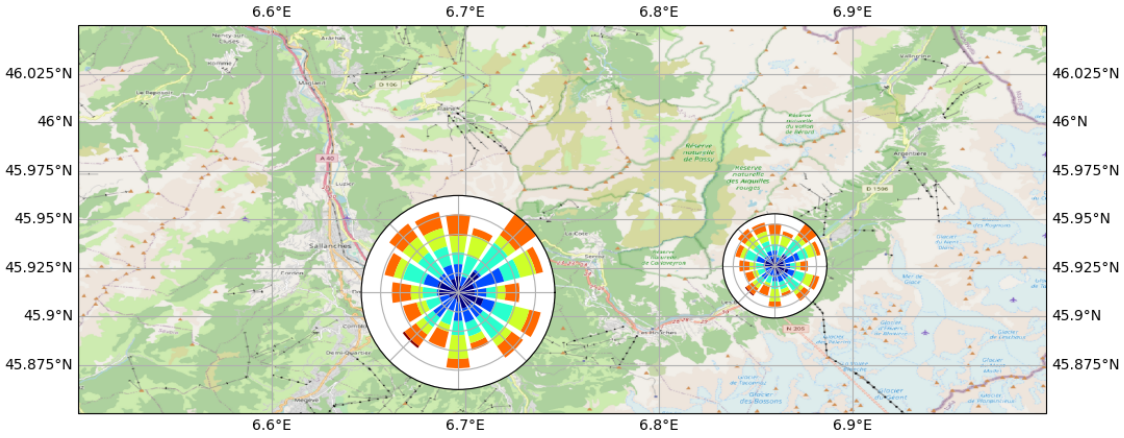
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid.inset_locator import inset_axes
import cartopy.crs as ccrs
```

(continues on next page)


0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
5
10
15
20
25
30

(continued from previous page)

```
import cartopy.io.img_tiles as cimgt

import windrose

ws = np.random.random(500) * 6
wd = np.random.random(500) * 360

minlon, maxlon, minlat, maxlat = (6.5, 7.0, 45.85, 46.05)

proj = ccrs.PlateCarree()
fig = plt.figure(figsize=(12, 6))
# Draw main ax on top of which we will add windroses
main_ax = fig.add_subplot(1, 1, 1, projection=proj)
main_ax.set_extent([minlon, maxlon, minlat, maxlat], crs=proj)
main_ax.gridlines(draw_labels=True)
main_ax.coastlines()

request = cimgt.OSM()
main_ax.add_image(request, 12)

# Coordinates of the station we were measuring windspeed
cham_lon, cham_lat = (6.8599, 45.9259)
passy_lon, passy_lat = (6.7, 45.9159)

# Inset axe it with a fixed size
wrax_cham = inset_axes(main_ax,
        width=1,                                # size in inches
        height=1,                               # size in inches
        loc='center',                           # center bbox at given position
        bbox_to_anchor=(cham_lon, cham_lat), # position of the axe
        bbox_transform=main_ax.transData,    # use data coordinate (not axe␣
↪coordinate)
        axes_class=windrose.WindroseAxes,    # specify the class of the axe
        )

# Inset axe with size relative to main axe
height_deg = 0.1
wrax_passy = inset_axes(main_ax,
        width="100%",                           # size in % of bbox
        height="100%",                          # size in % of bbox
        #loc='center',  # don't know why, but this doesn't work.
        # specify the center lon and lat of the plot, and size in degree
        bbox_to_anchor=(passy_lon-height_deg/2, passy_lat-height_deg/2, height_deg,␣
↪height_deg),
        bbox_transform=main_ax.transData,
        axes_class=windrose.WindroseAxes,
        )

wrax_cham.bar(wd, ws)
wrax_passy.bar(wd, ws)
for ax in [wrax_cham, wrax_passy]:
  ax.tick_params(labelleft=False, labelbottom=False)
```


6.6°E
6.7°E
6.8°E
6.9°E
46.025°N
46°N
45.975°N
45.95°N
45.925°N
45.9°N
45.875°N

CHAPTER 4

Functional API

Instead of using object oriented approach like previously shown, some "shortcut" functions have been defined: `wrbox`, `wrbar`, `wrcontour`, `wrcontourf`, `wrpdf`. See unit tests.

CHAPTER 5

Pandas support

windrose not only supports Numpy arrays. It also supports also Pandas DataFrame. `plot_windrose` function provides most of plotting features previously shown.

```
from windrose import plot_windrose
N = 500
ws = np.random.random(N) * 6
wd = np.random.random(N) * 360
df = pd.DataFrame({'speed': ws, 'direction': wd})
plot_windrose(df, kind='contour', bins=np.arange(0.01,8,1), cmap=cm.hot, lw=3)
```

Mandatory:

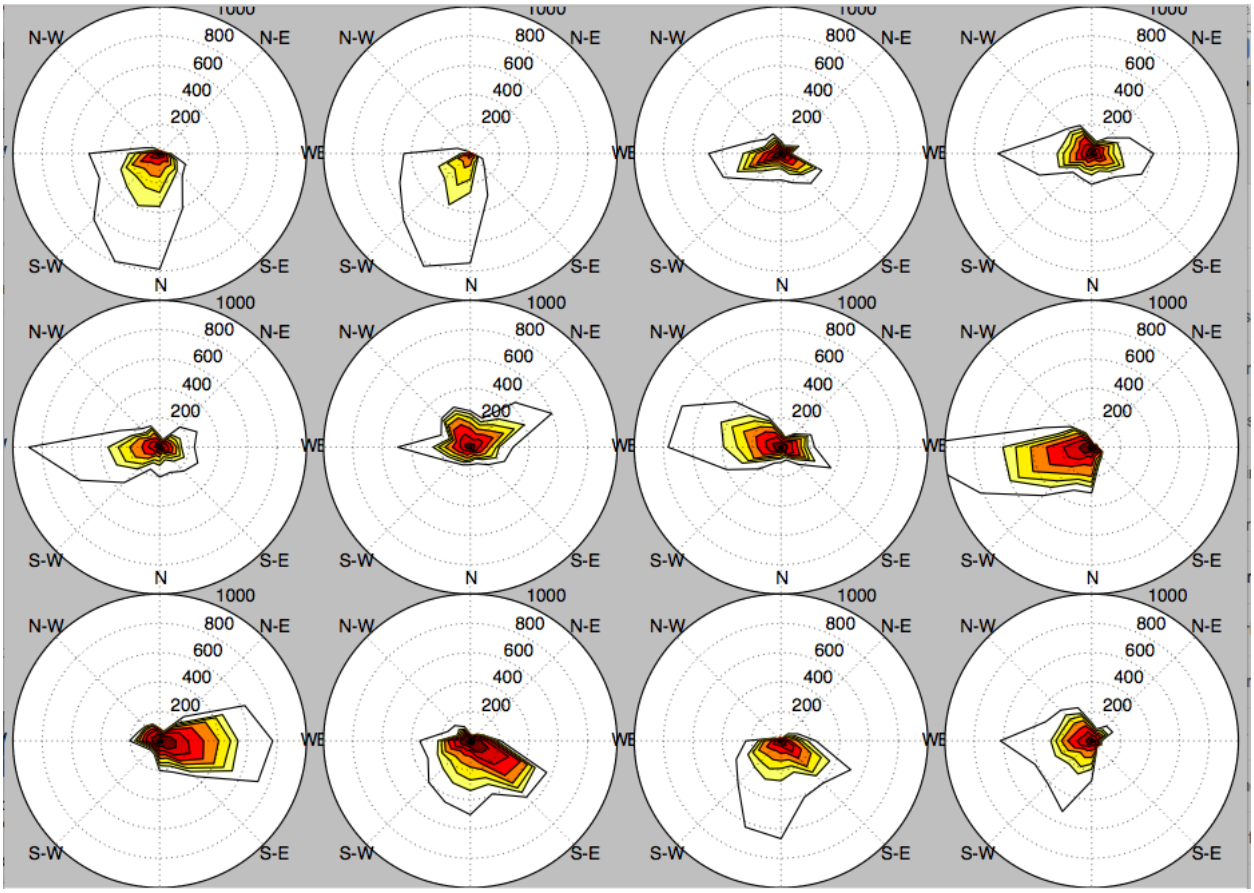
- `df`: Pandas DataFrame with `DateTimeIndex` as index and at least 2 columns (`'speed'` and `'direction'`).

Optional:

- `kind` : kind of plot (might be either, `'contour'`, `'contourf'`, `'bar'`, `'box'`, `'pdf'`)
- `var_name` : name of var column name ; default value is `VAR_DEFAULT='speed'`
- `direction_name` : name of direction column name ; default value is `DIR_DEFAULT='direction'`
- `clean_flag` : cleanup data flag (remove data points with `NaN`, `var=0`) before plotting ; default value is `True`.

CHAPTER 6

Subplots

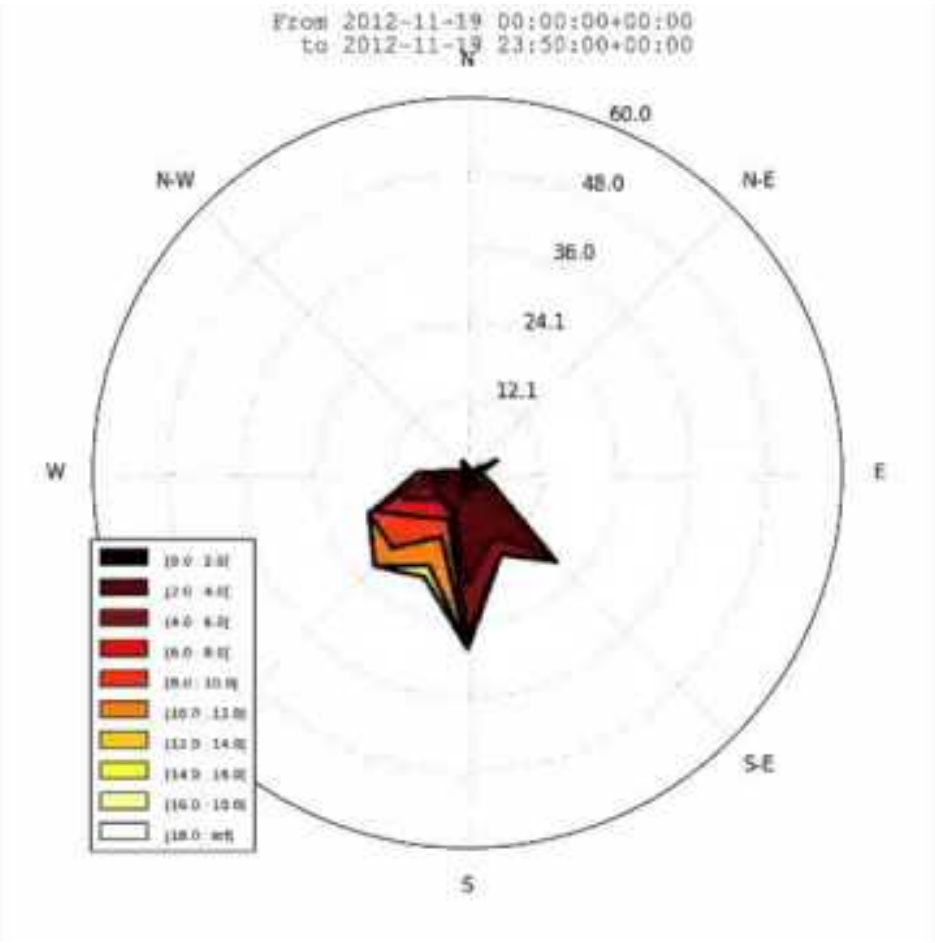


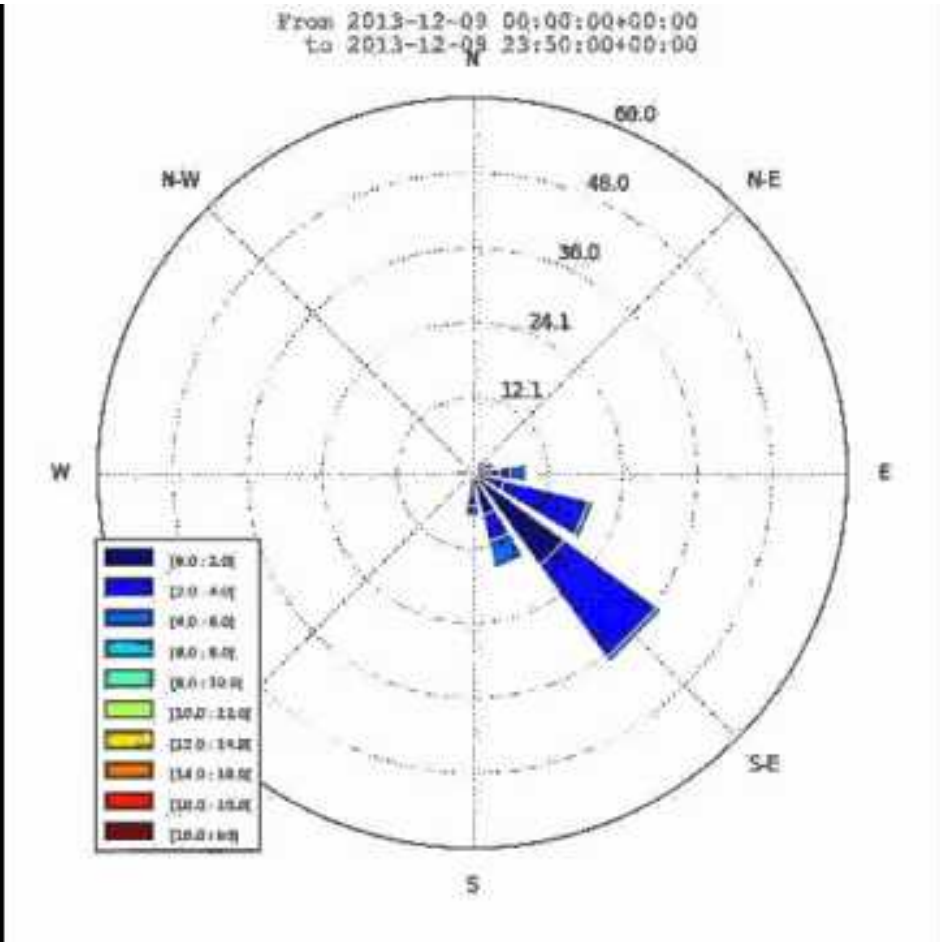
CHAPTER 7

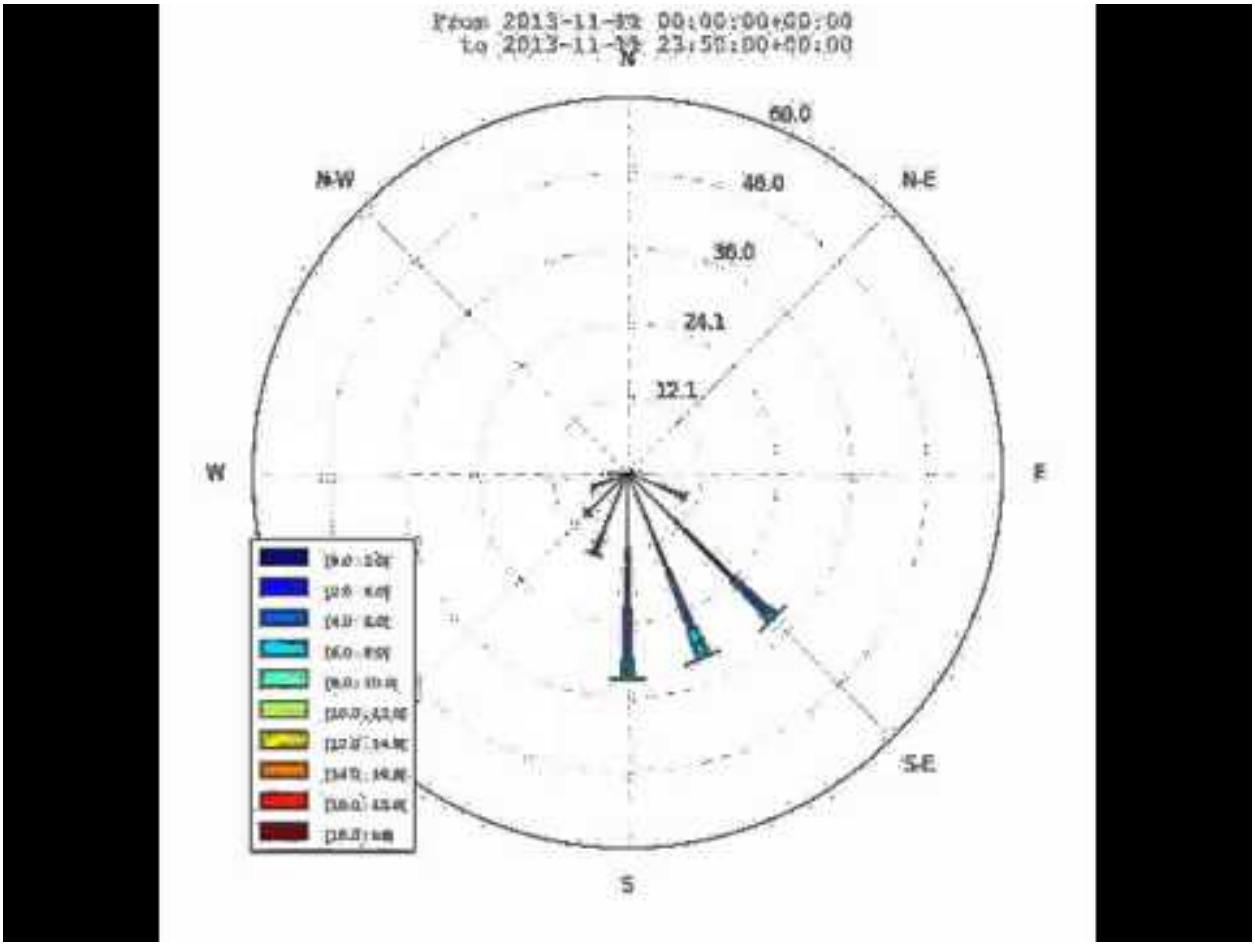
Video export

A video of plots can be exported. A playlist of videos is available at https://www.youtube.com/playlist?list=PLE9hIvV5BUzsQ4EPBDnJucgmmZ85D_b-W

See:


From 2012-11-19 00:00:00+00:00
to 2012-11-19 23:50:00+00:00
N
N-E
E
S-E
S
W
N-W
60.0
48.0
36.0
24.1
12.1


From 2013-12-09 00:00:00+00:00
to 2013-12-09 23:50:00+00:00
N
N-E
E
S-E
S
W
N-W
60.0
48.0
36.0
24.1
12.1



Source code

This is just a sample for now. API for video need to be created.

Use:

```
$ python samples/example_animate.py --help
```

to display command line interface usage.

CHAPTER 8

Development

You can help to develop this library.

8.1 Issues

You can submit issues using https://github.com/python-windrose/windrose/issues

8.2 Clone

You can clone repository to try to fix issues yourself using:

```
$ git clone https://github.com/python-windrose/windrose.git
```

8.3 Run unit tests

Run all unit tests

```
$ pytest -vv tests
```

Run a given test

```
$ pytest -vv tests/test_windrose.py::test_windrose_np_plot_and_pd_plot
```

8.4 Install development version

```
$ python setup.py install
```

or

```
$ sudo pip install git+https://github.com/python-windrose/windrose.git
```

8.5 Collaborating

- Fork repository
- Create a branch which fix a given issue
- Submit pull requests

CHAPTER 9

API

class windrose.**WindAxes**(**args*, ***kwargs*)

static from_ax(*ax=None*, *fig=None*, **args*, ***kwargs*)

pdf(*var*, *bins=None*, *Nx=100*, *bar_color='b'*, *plot_color='g'*, *Nbins=10*, **args*, ***kwargs*)
Draw probability density function and return Weibull distribution parameters

class windrose.**WindAxesFactory**
Factory class to create WindroseAxes or WindAxes

static create(*typ*, *ax=None*, **args*, ***kwargs*)
Create

Mandatory:

Parameters

- **typ** (*string, 'windroseaxes' or 'windaxes'*) –

  Type of axes to create

  - windroseaxes : a WindroseAxes axe
  - windaxe : a WindAxes axe

- **ax** (*matplotlib.Axes, optional*) – A matplotlib axe

class windrose.**WindroseAxes**(**args*, ***kwargs*)
Create a windrose axes

bar(*direction*, *var*, ***kwargs*)
Plot a windrose in bar mode. For each var bins and for each sector, a colored bar will be draw on the axes.

Parameters

- **direction** (*1D array*) – directions the wind blows from, North centred
- **var** (*1D array*) – values of the variable to compute. Typically the wind speeds.

Other Parameters

- **nsector** (*integer, optional*) – number of sectors used to compute the windrose table. If not set, nsectors=16, then each sector will be 360/16=22.5°, and the resulting computed table will be aligned with the cardinals points.
- **bins** (*1D array or integer, optional*) – number of bins, or a sequence of bins variable. If not set, bins=6 between min(*var*) and max(*var*).
- **blowto** (*bool, optional.*) – if True, the windrose will be pi rotated, to show where the wind blow to (useful for pollutant rose).
- **colors** (*string or tuple, optional*) – one string color ('k' or 'black'), in this case all bins will be plotted in this color; a tuple of matplotlib color args (string, float, rgb, etc), different levels will be plotted in different colors in the order specified.
- **cmap** (a cm Colormap instance from `matplotlib.cm`, optional.) – if cmap == None and colors == None, a default Colormap is used.
- **edgecolor** (*string, optional*) – The string color each edge box will be plotted. Default : no edgecolor
- **opening** (*float, optional*) – between 0.0 and 1.0, to control the space between each sector (1.0 for no space)
- **calm_limit** (*float, optional*) – Calm limit for the var parameter. If not None, a centered red circle will be draw for representing the calms occurrences and all data below this value will be removed from the computation.

box(*direction*, *var*, ***kwargs*)

Plot a windrose in proportional box mode. For each var bins and for each sector, a colored box will be draw on the axes.

Parameters

- **direction** (*1D array*) – directions the wind blows from, North centred
- **var** (*1D array*) – values of the variable to compute. Typically the wind speeds

Other Parameters

- **nsector** (*integer, optional*) – number of sectors used to compute the windrose table. If not set, nsectors=16, then each sector will be 360/16=22.5°, and the resulting computed table will be aligned with the cardinals points.
- **bins** (*1D array or integer, optional*) – number of bins, or a sequence of bins variable. If not set, bins=6 between min(*var*) and max(*var*).
- **blowto** (*bool, optional*) – If True, the windrose will be pi rotated, to show where the wind blow to (useful for pollutant rose).
- **colors** (*string or tuple, optional*) – one string color ('k' or 'black'), in this case all bins will be plotted in this color; a tuple of matplotlib color args (string, float, rgb, etc), different levels will be plotted in different colors in the order specified.
- **cmap** (a cm Colormap instance from `matplotlib.cm`, optional) – if cmap == None and colors == None, a default Colormap is used.
- **edgecolor** (*string, optional*) – The string color each edge bar will be plotted. Default : no edgecolor
- **calm_limit** (*float, optional*) – Calm limit for the var parameter. If not None, a centered red circle will be draw for representing the calms occurrences and all data below this value will be removed from the computation.

cla()
Clear the current axes

contour(*direction*, *var*, ***kwargs*)
Plot a windrose in linear mode. For each var bins, a line will be draw on the axes, a segment between each sector (center to center). Each line can be formatted (color, width, ...) like with standard plot pylab command.

Parameters

- **direction** (*1D array*) – directions the wind blows from, North centred
- **var** (*1D array*) – values of the variable to compute. Typically the wind speeds.

Other Parameters

- **sector** (*integer, optional*) – number of sectors used to compute the windrose table. If not set, nsectors=16, then each sector will be 360/16=22.5°, and the resulting computed table will be aligned with the cardinals points.
- **bins** (*1D array or integer, optional*) – number of bins, or a sequence of bins variable. If not set, bins=6, then bins=linspace(min(var), max(var), 6)
- **blowto** (*bool, optional*) – If True, the windrose will be pi rotated, to show where the wind blow to (useful for pollutant rose).
- **colors** (*string or tuple, optional*) – one string color ('k' or 'black'), in this case all bins will be plotted in this color; a tuple of matplotlib color args (string, float, rgb, etc), different levels will be plotted in different colors in the order specified.
- **cmap** (a cm Colormap instance from `matplotlib.cm`, optional) – if cmap == None and colors == None, a default Colormap is used.
- **calm_limit** (*float, optional*) – Calm limit for the var parameter. If not None, a centered red circle will be draw for representing the calms occurrences and all data below this value will be removed from the computation.
- **others kwargs** – Any supported argument of `matplotlib.pyplot.plot`

contourf(*direction*, *var*, ***kwargs*)
Plot a windrose in filled mode. For each var bins, a line will be draw on the axes, a segment between each sector (center to center). Each line can be formatted (color, width, ...) like with standard plot pylab command.

Parameters

- **direction** (*1D array*) – directions the wind blows from, North centred
- **var** (*1D array*) – values of the variable to compute. Typically the wind speeds

Other Parameters

- **nsector** (*integer, optional*) – number of sectors used to compute the windrose table. If not set, nsectors=16, then each sector will be 360/16=22.5°, and the resulting computed table will be aligned with the cardinals points.
- **bins** (*1D array or integer, optional*) – number of bins, or a sequence of bins variable. If not set, bins=6, then bins=linspace(min(*var*), max(*var*), 6)
- **blowto** (*bool, optional*) – If True, the windrose will be pi rotated, to show where the wind blow to (useful for pollutant rose).

- **colors** (*string or tuple, optional*) – one string color ('k' or 'black'), in this case all bins will be plotted in this color; a tuple of matplotlib color args (string, float, rgb, etc), different levels will be plotted in different colors in the order specified.
- **cmap** (a cm Colormap instance from `matplotlib.cm`, optional) – if cmap == None and colors == None, a default Colormap is used.
- **calm_limit** (*float, optional*) – Calm limit for the var parameter. If not None, a centered red circle will be draw for representing the calms occurrences and all data below this value will be removed from the computation.
- **others kwargs** – Any supported argument of `matplotlib.pyplot.plot`

static from_ax(*ax=None*, *fig=None*, *rmax=None*, *theta_labels=None*, *rect=None*, **args*, ***kwargs*)

Return a WindroseAxes object for the figure *fig*.

legend(*loc='lower left'*, *decimal_places=1*, *units=None*, ***kwargs*)

Sets the legend location and her properties.

Parameters

- **loc** (*int, string or pair of floats, default: 'lower left'*) – see `matplotlib.pyplot.legend`.
- **decimal_places** (*int, default 1*) – The decimal places of the formatted legend
- **units** (*str, default None*) –

Other Parameters

- **isaxes** (*boolean, default True*) – whether this is an axes legend
- **prop** (*FontProperties(size='smaller')*) – the font property
- **borderpad** (*float*) – the fractional whitespace inside the legend border
- **shadow** (*boolean*) – if True, draw a shadow behind legend
- **labelspacing** (*float, 0.005*) – the vertical space between the legend entries
- **handlelenght** (*float, 0.05*) – the length of the legend lines
- **handletextsep** (*float, 0.02*) – the space between the legend line and legend text
- **borderaxespad** (*float, 0.02*) – the border between the axes and legend edge
- **kwarg** – Every other kwarg argument supported by `matplotlib.pyplot.legend`

name = 'windrose'

set_legend(***pyplot_arguments*)

set_radii_angle(***kwargs*)

Set the radii labels angle

windrose.**clean**(*direction*, *var*, *index=False*)

Remove nan and var=0 values in the two arrays if a var (wind speed) is nan or equal to 0, this data is removed from var array but also from dir array if a direction is nan, data is also removed from both array

windrose.**clean_df**(*df*, *var='speed'*, *direction='direction'*)

Remove nan and var=0 values in the DataFrame if a var (wind speed) is nan or equal to 0, this row is removed from DataFrame if a direction is nan, this row is also removed from DataFrame

windrose.**histogram**(*direction*, *var*, *bins*, *nsector*, *normed=False*, *blowto=False*)

Returns an array where, for each sector of wind (centred on the north), we have the number of time the wind comes with a particular var (speed, pollutant concentration, ...).

Parameters

- **direction** (*1D array*) – directions the wind blows from, North centred
- **var** (*1D array*) – values of the variable to compute. Typically the wind speeds
- **bins** (*list*) – list of var category against we're going to compute the table
- **nsector** (*integer*) – number of sectors

Other Parameters

- **normed** (*boolean, default False*) – The resulting table is normed in percent or not.
- **blowto** (*boolean, default False*) – Normally a windrose is computed with directions as wind blows from. If true, the table will be reversed (useful for pollutantrose)

windrose.**plot_windrose**(*direction_or_df*, *var=None*, *kind='contour'*, *var_name='speed'*, *direction_name='direction'*, *by=None*, *rmax=None*, ***kwargs*)

windrose.**plot_windrose_df**(*df*, *kind='contour'*, *var_name='speed'*, *direction_name='direction'*, *by=None*, *rmax=None*, ***kwargs*)

windrose.**plot_windrose_np**(*direction*, *var*, *kind='contour'*, *clean_flag=True*, *by=None*, *rmax=None*, ***kwargs*)

windrose.**wrbar**(*direction*, *var*, *ax=None*, *rmax=None*, ***kwargs*)

Plot a windrose in bar mode. For each var bins and for each sector, a colored bar will be draw on the axes.

Parameters

- **direction** (*1D array*) – directions the wind blows from, North centred
- **var** (*1D array*) – values of the variable to compute. Typically the wind speeds.

Other Parameters

- **nsector** (*integer, optional*) – number of sectors used to compute the windrose table. If not set, nsectors=16, then each sector will be 360/16=22.5°, and the resulting computed table will be aligned with the cardinals points.
- **bins** (*1D array or integer, optional*) – number of bins, or a sequence of bins variable. If not set, bins=6 between min(*var*) and max(*var*).
- **blowto** (*bool, optional.*) – if True, the windrose will be pi rotated, to show where the wind blow to (useful for pollutant rose).
- **colors** (*string or tuple, optional*) – one string color ('k' or 'black'), in this case all bins will be plotted in this color; a tuple of matplotlib color args (string, float, rgb, etc), different levels will be plotted in different colors in the order specified.
- **cmap** (a cm Colormap instance from matplotlib.cm, optional.) – if cmap == None and colors == None, a default Colormap is used.
- **edgecolor** (*string, optional*) – The string color each edge box will be plotted. Default : no edgecolor
- **opening** (*float, optional*) – between 0.0 and 1.0, to control the space between each sector (1.0 for no space)

- **calm_limit** (*float, optional*) – Calm limit for the var parameter. If not None, a centered red circle will be draw for representing the calms occurrences and all data below this value will be removed from the computation.

windrose.**wrbox**(*direction*, *var*, *ax=None*, *rmax=None*, ***kwargs*)

Plot a windrose in proportional box mode. For each var bins and for each sector, a colored box will be draw on the axes.

Parameters

- **direction** (*1D array*) – directions the wind blows from, North centred
- **var** (*1D array*) – values of the variable to compute. Typically the wind speeds

Other Parameters

- **nsector** (*integer, optional*) – number of sectors used to compute the windrose table. If not set, nsectors=16, then each sector will be 360/16=22.5°, and the resulting computed table will be aligned with the cardinals points.
- **bins** (*1D array or integer, optional*) – number of bins, or a sequence of bins variable. If not set, bins=6 between min(*var*) and max(*var*).
- **blowto** (*bool, optional*) – If True, the windrose will be pi rotated, to show where the wind blow to (useful for pollutant rose).
- **colors** (*string or tuple, optional*) – one string color ('k' or 'black'), in this case all bins will be plotted in this color; a tuple of matplotlib color args (string, float, rgb, etc), different levels will be plotted in different colors in the order specified.
- **cmap** (a cm Colormap instance from `matplotlib.cm`, optional) – if cmap == None and colors == None, a default Colormap is used.
- **edgecolor** (*string, optional*) – The string color each edge bar will be plotted. Default : no edgecolor
- **calm_limit** (*float, optional*) – Calm limit for the var parameter. If not None, a centered red circle will be draw for representing the calms occurrences and all data below this value will be removed from the computation.

windrose.**wrcontour**(*direction*, *var*, *ax=None*, *rmax=None*, ***kwargs*)

Plot a windrose in linear mode. For each var bins, a line will be draw on the axes, a segment between each sector (center to center). Each line can be formatted (color, width, ...) like with standard plot pylab command.

Parameters

- **direction** (*1D array*) – directions the wind blows from, North centred
- **var** (*1D array*) – values of the variable to compute. Typically the wind speeds.

Other Parameters

- **sector** (*integer, optional*) – number of sectors used to compute the windrose table. If not set, nsectors=16, then each sector will be 360/16=22.5°, and the resulting computed table will be aligned with the cardinals points.
- **bins** (*1D array or integer, optional*) – number of bins, or a sequence of bins variable. If not set, bins=6, then bins=linspace(min(var), max(var), 6)
- **blowto** (*bool, optional*) – If True, the windrose will be pi rotated, to show where the wind blow to (useful for pollutant rose).

- **colors** (*string or tuple, optional*) – one string color ('k' or 'black'), in this case all bins will be plotted in this color; a tuple of matplotlib color args (string, float, rgb, etc), different levels will be plotted in different colors in the order specified.
- **cmap** (a cm Colormap instance from `matplotlib.cm`, optional) – if cmap == None and colors == None, a default Colormap is used.
- **calm_limit** (*float, optional*) – Calm limit for the var parameter. If not None, a centered red circle will be draw for representing the calms occurrences and all data below this value will be removed from the computation.
- **others kwargs** – Any supported argument of `matplotlib.pyplot.plot`

`windrose.`**`wrcontourf`**(*direction*, *var*, *ax=None*, *rmax=None*, ***kwargs*)

Plot a windrose in filled mode. For each var bins, a line will be draw on the axes, a segment between each sector (center to center). Each line can be formatted (color, width, ...) like with standard plot pylab command.

Parameters

- **`direction`** (*`1D array`*) – directions the wind blows from, North centred
- **`var`** (*`1D array`*) – values of the variable to compute. Typically the wind speeds

Other Parameters

- **nsector** (*integer, optional*) – number of sectors used to compute the windrose table. If not set, nsectors=16, then each sector will be 360/16=22.5°, and the resulting computed table will be aligned with the cardinals points.
- **bins** (*1D array or integer, optional*) – number of bins, or a sequence of bins variable. If not set, bins=6, then bins=linspace(min(*var*), max(*var*), 6)
- **blowto** (*bool, optional*) – If True, the windrose will be pi rotated, to show where the wind blow to (useful for pollutant rose).
- **colors** (*string or tuple, optional*) – one string color ('k' or 'black'), in this case all bins will be plotted in this color; a tuple of matplotlib color args (string, float, rgb, etc), different levels will be plotted in different colors in the order specified.
- **cmap** (a cm Colormap instance from `matplotlib.cm`, optional) – if cmap == None and colors == None, a default Colormap is used.
- **calm_limit** (*float, optional*) – Calm limit for the var parameter. If not None, a centered red circle will be draw for representing the calms occurrences and all data below this value will be removed from the computation.
- **others kwargs** – Any supported argument of `matplotlib.pyplot.plot`

`windrose.`**`wrpdf`**(*var*, *bins=None*, *Nx=100*, *bar_color='b'*, *plot_color='g'*, *Nbins=10*, *ax=None*, *rmax=None*, **args*, ***kwargs*)

Draw probability density function and return Weibull distribution parameters

`windrose.`**`wrscatter`**(*direction*, *var*, *ax=None*, *rmax=None*, **args*, ***kwargs*)

Draw scatter plot

A windrose, also known as a polar rose plot, is a special diagram for representing the distribution of meteorological data, typically wind speeds by class and direction. This is a simple module for the matplotlib python library, which requires numpy for internal computation.

Original code forked from: - windrose 1.4 by Lionel Roubeyrie lionel.roubeyrie@gmail.com http://youarealegend.blogspot.com/search/label/windrose

https://docs.github.com/en/pull-requests/collaborating-with-pull-requests

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Python Module Index

W

Index