

---

# **windrose Documentation**

**Lionel Roubeyrie & Sebastien Celles**

**Jun 10, 2019**



---

## Contents:

---

<b>1</b>	<b>Install</b>	<b>3</b>
1.1	Requirements . . . . .	3
1.2	Install latest release version via pip . . . . .	3
1.3	Install latest development version . . . . .	3
<b>2</b>	<b>Notebook example</b>	<b>5</b>
<b>3</b>	<b>Script example</b>	<b>7</b>
3.1	A stacked histogram with normed (displayed in percent) results . . . . .	7
3.2	Another stacked histogram representation, not normed, with bins limits . . . . .	7
3.3	A windrose in filled representation, with a controled colormap . . . . .	9
3.4	Same as above, but with contours over each filled region. . . . .	9
3.5	...or without filled regions . . . . .	9
3.6	probability density function (pdf) and fitting Weibull distribution . . . . .	12
<b>4</b>	<b>Functional API</b>	<b>15</b>
<b>5</b>	<b>Pandas support</b>	<b>17</b>
<b>6</b>	<b>Subplots</b>	<b>19</b>
<b>7</b>	<b>Video export</b>	<b>21</b>
<b>8</b>	<b>Development</b>	<b>23</b>
8.1	Issues . . . . .	23
8.2	Clone . . . . .	23
8.3	Run unit tests . . . . .	23
8.4	Install development version . . . . .	24
8.5	Collaborating . . . . .	24
<b>9</b>	<b>API</b>	<b>25</b>
<b>10</b>	<b>Indices and tables</b>	<b>33</b>
	<b>Python Module Index</b>	<b>35</b>
	<b>Index</b>	<b>37</b>







### 1.1 Requirements

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

Option libraries:

- Pandas <http://pandas.pydata.org/> (to feed plot functions easily)
- Scipy <http://www.scipy.org/> (to fit data with Weibull distribution)
- ffmpeg <https://www.ffmpeg.org/> (to output video)
- click <http://click.pocoo.org/> (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:

- [http://nbviewer.ipython.org/github/python-windrose/windrose/blob/master/windrose\\_sample\\_random.ipynb](http://nbviewer.ipython.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.pyplot 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

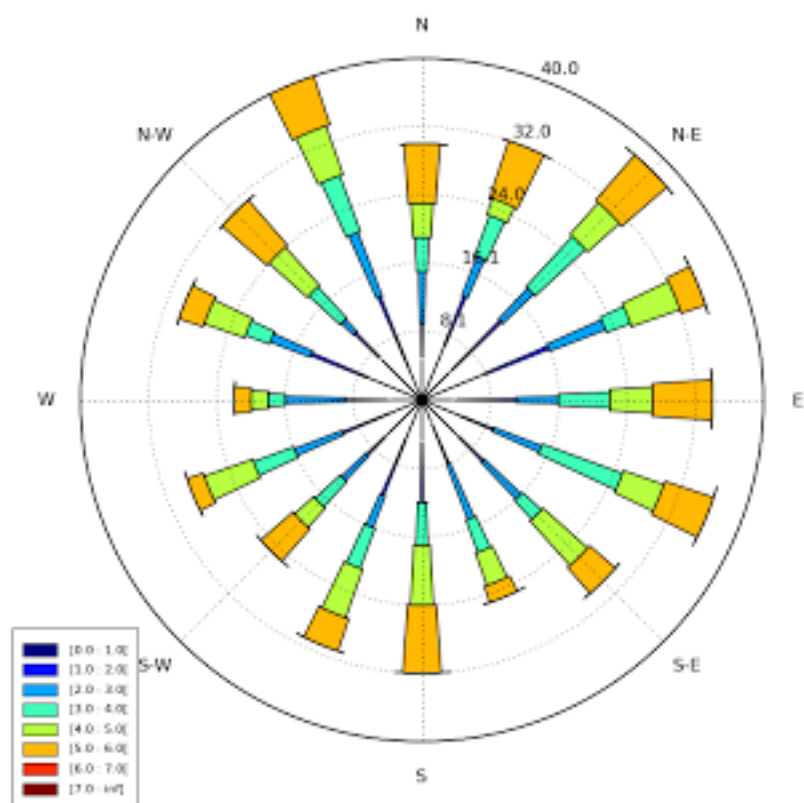
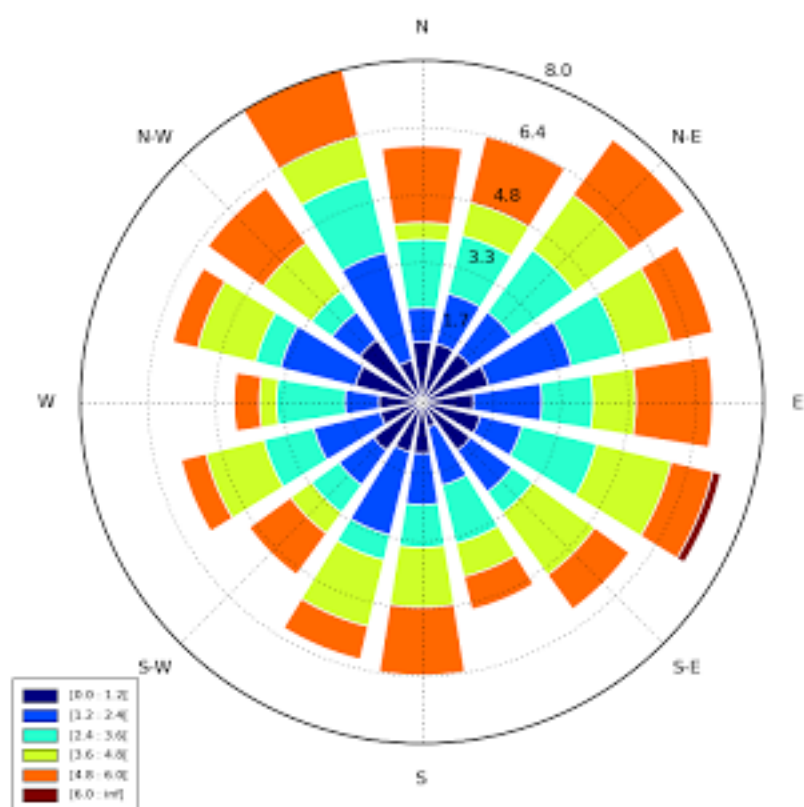
ws = np.random.random(500) * 6
wd = np.random.random(500) * 360
```

### 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()
```

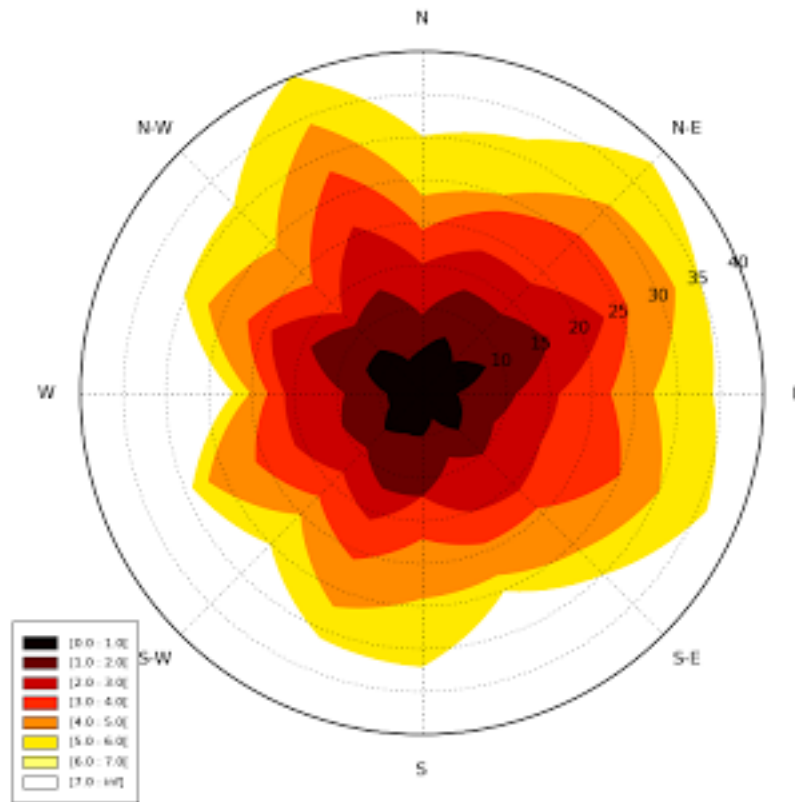
### 3.2 Another stacked histogram representation, not normed, with bins limits

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



### 3.3 A windrose in filled representation, with a controled colormap

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



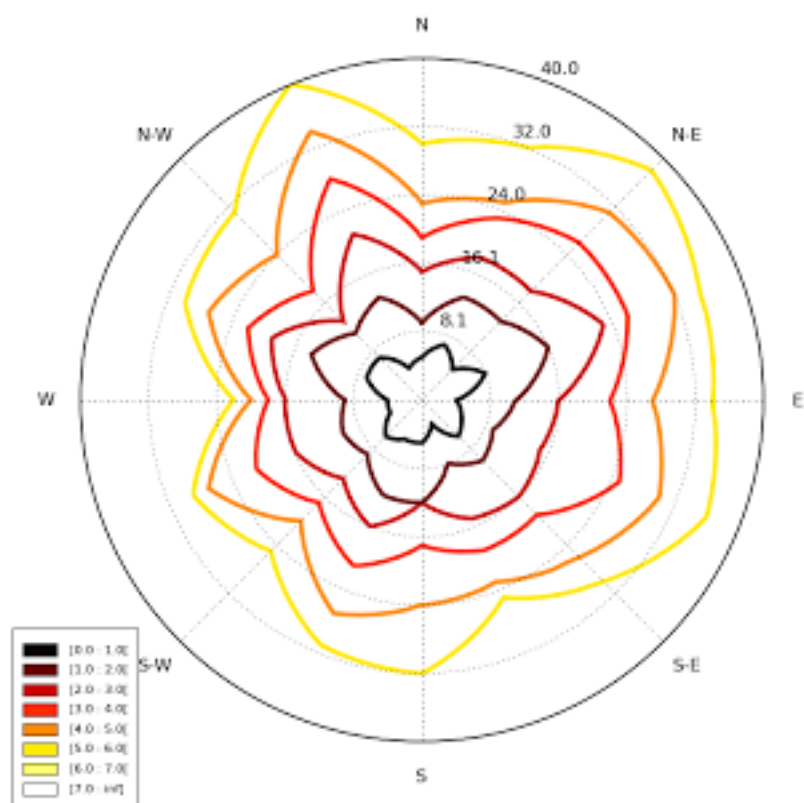
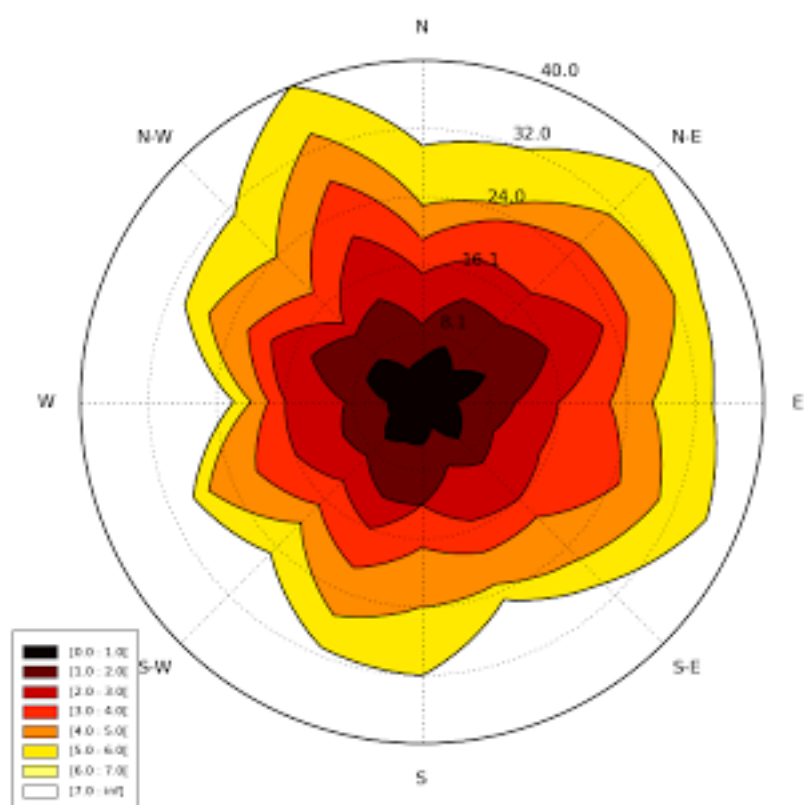
### 3.4 Same as above, but with contours over each filled region...

```
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 :



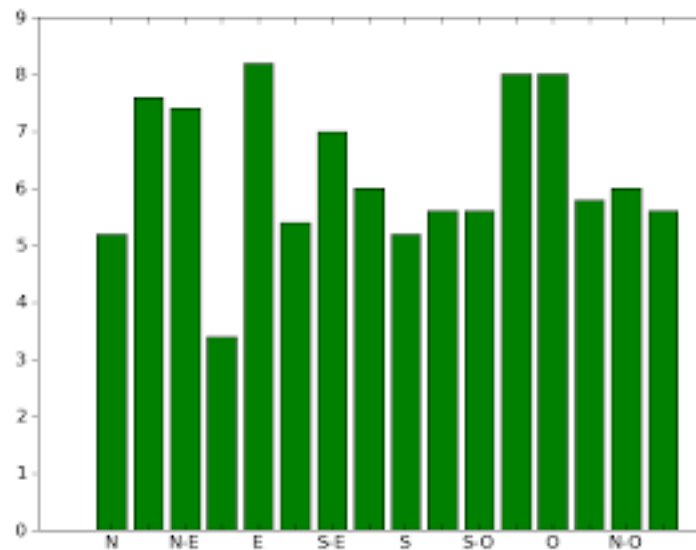
- `ax._info['bins']` : list of bins (limits) used for wind speeds. If not set in the call, bins will be set to 6 parts between wind speed min and max.
- `ax._info['dir']` : list of directions “bundaries” used to compute the distribution by wind direction sector. This can be set by the `nsector` parameter (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  $360/16=22.5^\circ$ , 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 (usefull 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

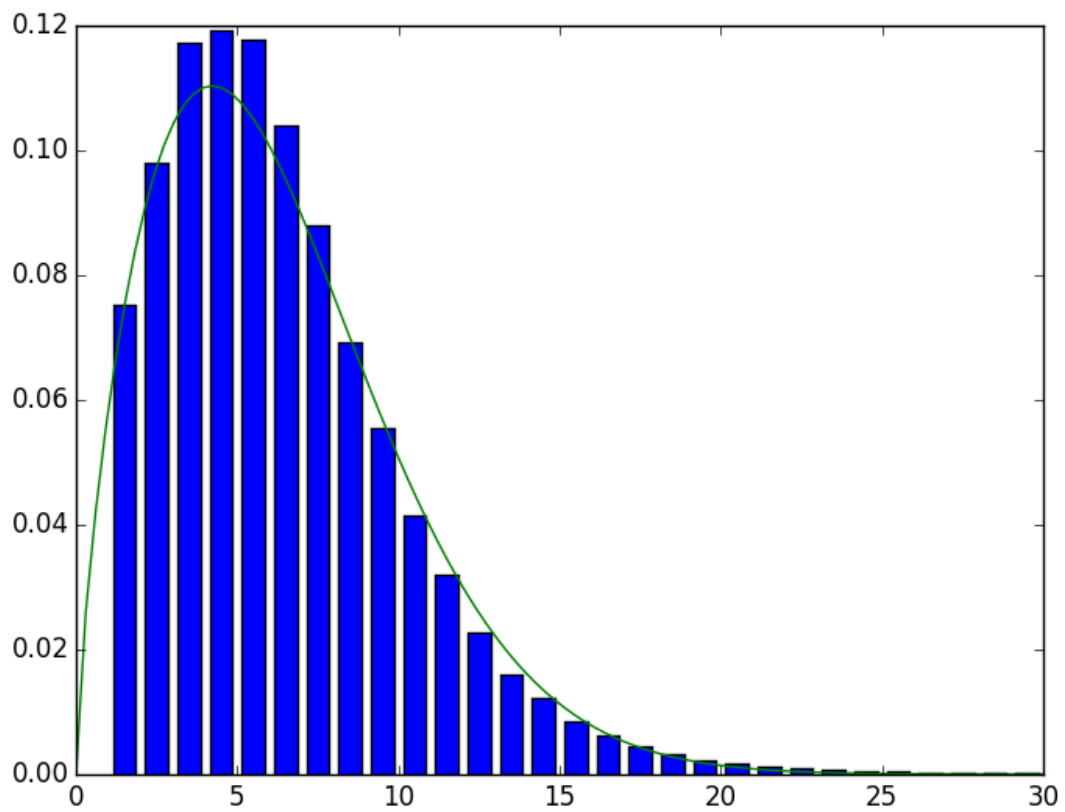
A probability density function can be plot using:

```
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)
```







## 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').

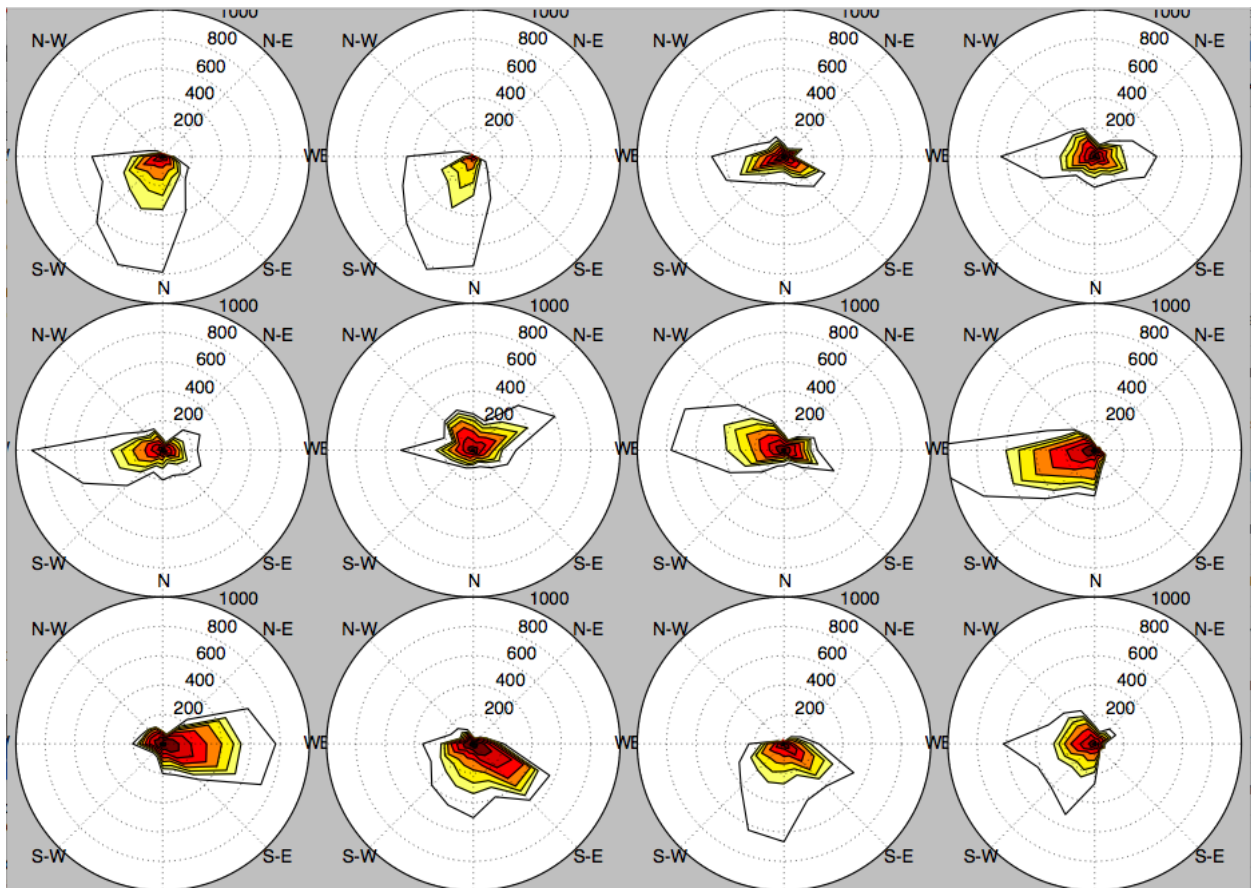
Optional:

- `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](https://www.youtube.com/playlist?list=PLE9hIvV5BUzsQ4EPBDnJucgmmZ85D_b-W)

See:

**|Video1| |Video2| |Video3|**

[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.



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

---

```

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  $360/16=22.5^\circ$ , 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 (usefull 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)

**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  $360/16=22.5^\circ$ , 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 (usefull 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

**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  $360/16=22.5^\circ$ , 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 (usefull 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.
- **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  $360/16=22.5^\circ$ , 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 (usefull 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.
- **others kwargs** – Any supported argument of matplotlib.pyplot.plot

**static from\_ax** (*ax=None, fig=None, rmax=None, \*args, \*\*kwargs*)

Return a WindroseAxes object for the figure fig.

**legend** (*loc='lower left', decimal\_places=1, \*\*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

#### 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
- **handlelength** (*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*) – Normaly a windrose is computed with directions as wind blows from. If true, the table will be reversed (usefull 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  $360/16=22.5^\circ$ , 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  $\min(\text{var})$  and  $\max(\text{var})$ .
- **blowto** (*bool, optional.*) – if True, the windrose will be pi rotated, to show where the wind blow to (usefull 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)

`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  $360/16=22.5^\circ$ , 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  $\min(\text{var})$  and  $\max(\text{var})$ .
- **blowto** (*bool, optional*) – If True, the windrose will be pi rotated, to show where the wind blow to (usefull 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

`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  $360/16=22.5^\circ$ , 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 (usefull 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.
- **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  $360/16=22.5^\circ$ , 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 (usefull 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.
- **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 datas, 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](mailto:lionel.roubeyrie@gmail.com) [lionel.roubeyrie@gmail.com](http://youarealegend.blogspot.fr/search/label/windrose) <http://youarealegend.blogspot.fr/search/label/windrose>

<https://help.github.com/categories/collaborating/>



## CHAPTER 10

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



**W**

windrose, [25](#)





## B

`bar()` (*windrose.WindroseAxes method*), 25  
`box()` (*windrose.WindroseAxes method*), 26

## C

`cla()` (*windrose.WindroseAxes method*), 26  
`clean()` (*in module windrose*), 28  
`clean_df()` (*in module windrose*), 28  
`contour()` (*windrose.WindroseAxes method*), 26  
`contourf()` (*windrose.WindroseAxes method*), 27  
`create()` (*windrose.WindAxesFactory static method*), 25

## F

`from_ax()` (*windrose.WindAxes static method*), 25  
`from_ax()` (*windrose.WindroseAxes static method*), 27

## H

`histogram()` (*in module windrose*), 28

## L

`legend()` (*windrose.WindroseAxes method*), 27

## N

`name` (*windrose.WindroseAxes attribute*), 28

## P

`pdf()` (*windrose.WindAxes method*), 25  
`plot_windrose()` (*in module windrose*), 28  
`plot_windrose_df()` (*in module windrose*), 28  
`plot_windrose_np()` (*in module windrose*), 29

## S

`set_legend()` (*windrose.WindroseAxes method*), 28  
`set_radii_angle()` (*windrose.WindroseAxes method*), 28

## W

`WindAxes` (*class in windrose*), 25

`WindAxesFactory` (*class in windrose*), 25  
`windrose` (*module*), 25  
`WindroseAxes` (*class in windrose*), 25  
`wrbar()` (*in module windrose*), 29  
`wrbox()` (*in module windrose*), 29  
`wrcontour()` (*in module windrose*), 30  
`wrcontourf()` (*in module windrose*), 30  
`wrpdf()` (*in module windrose*), 30  
`wrscatter()` (*in module windrose*), 31