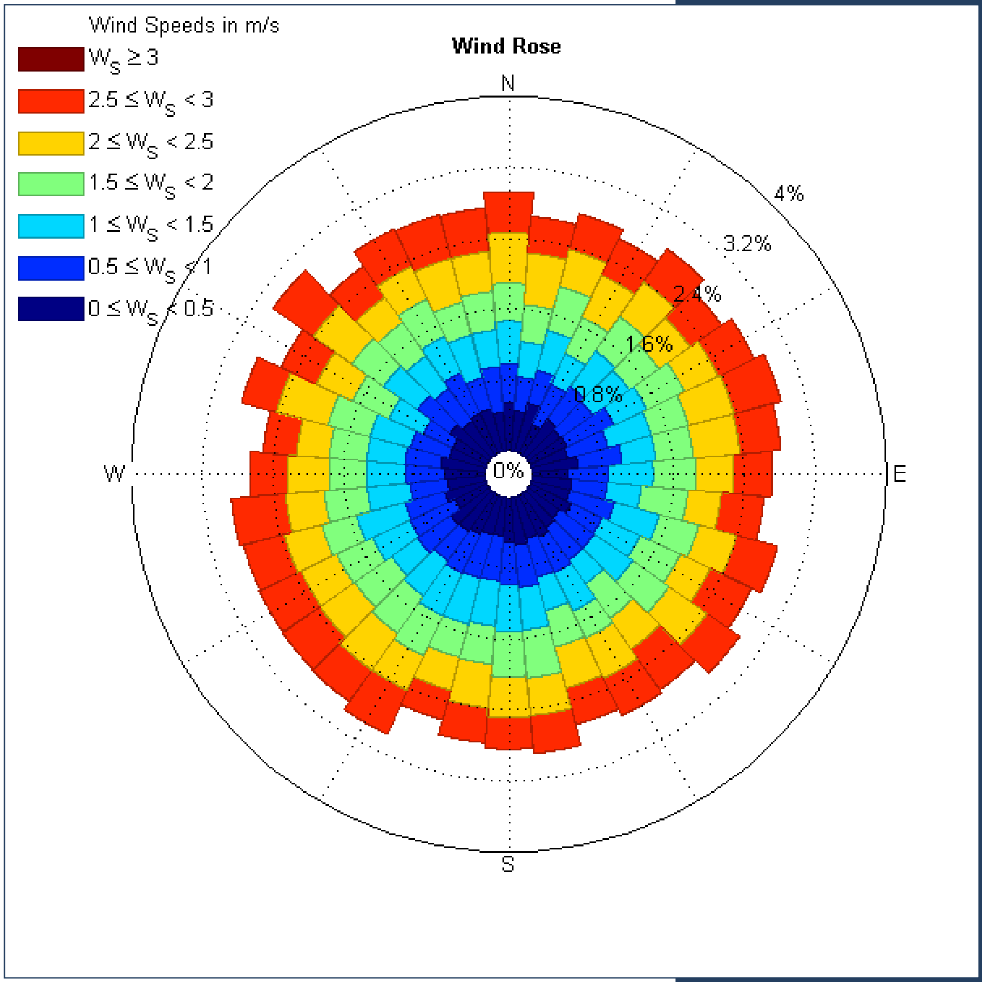
WindRose for Matlab



By Daniel Pereira

daniel.pereira.valades@gmail.com

14 March 2015

Content

Data

We start from some simple data which we want to be represented in a wind rose. These data could come from data measurement (temporal series, data collection, etc.)

```
clc; clear all; close all;
dir = mod(360*rand(20000,1),360);
spd = abs(rand(20000,1))*3;
```

Notes

This function can be called with many arguments at the same time. The obligatory input arguments are the wind directions and wind speeds (two different vectors). The following examples have been created in order to show the effect of a particular command, but all of them can be combined in the function call, within a structure, a cell array or the common call.

The following three samples show the different ways of calling the function, giving the same result.

1. With options in a cell array:

```
Options = {'anglenorth',0,'angleeast',90,'labels',{'N (0°)','S (180°)','E (90°)','W
(270°)'},'freqlabelangle',45};
[figure_handle,count,speeds,directions,Table] = WindRose(dir,spd,Options);
close all; clear Options;
```

2. With options in a structure:

```
Options.AngleNorth     = 0;
Options.AngleEast      = 90;
Options.Labels         = {'N (0°)','S (180°)','E (90°)','W (270°)'};
Options.FreqLabelAngle = 45;
[figure_handle,count,speeds,directions,Table] = WindRose(dir,spd,Options);
close all;
```

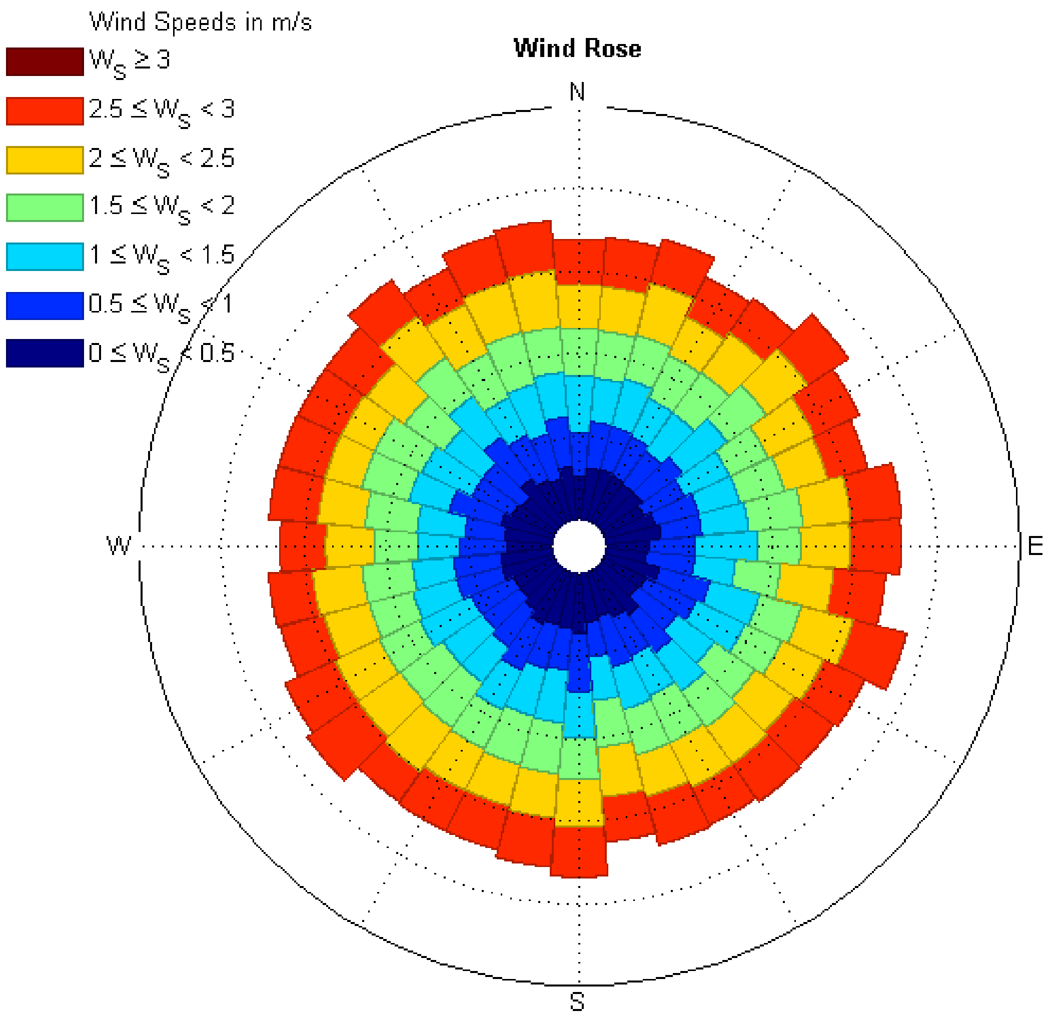
3. Usual calling:

```
[figure_handle,count,speeds,directions,Table] =
WindRose(dir,spd,'anglenorth',0,'angleeast',90,'labels',{'N (0°)','S (180°)','E
(90°)','W (270°)'},'freqlabelangle',45);
close all; clear Options;
```

Simple usage

Just plotting patches in each direction:

```
[figure_handle,count,speeds,directions,Table] = WindRose(dir,spd);
```

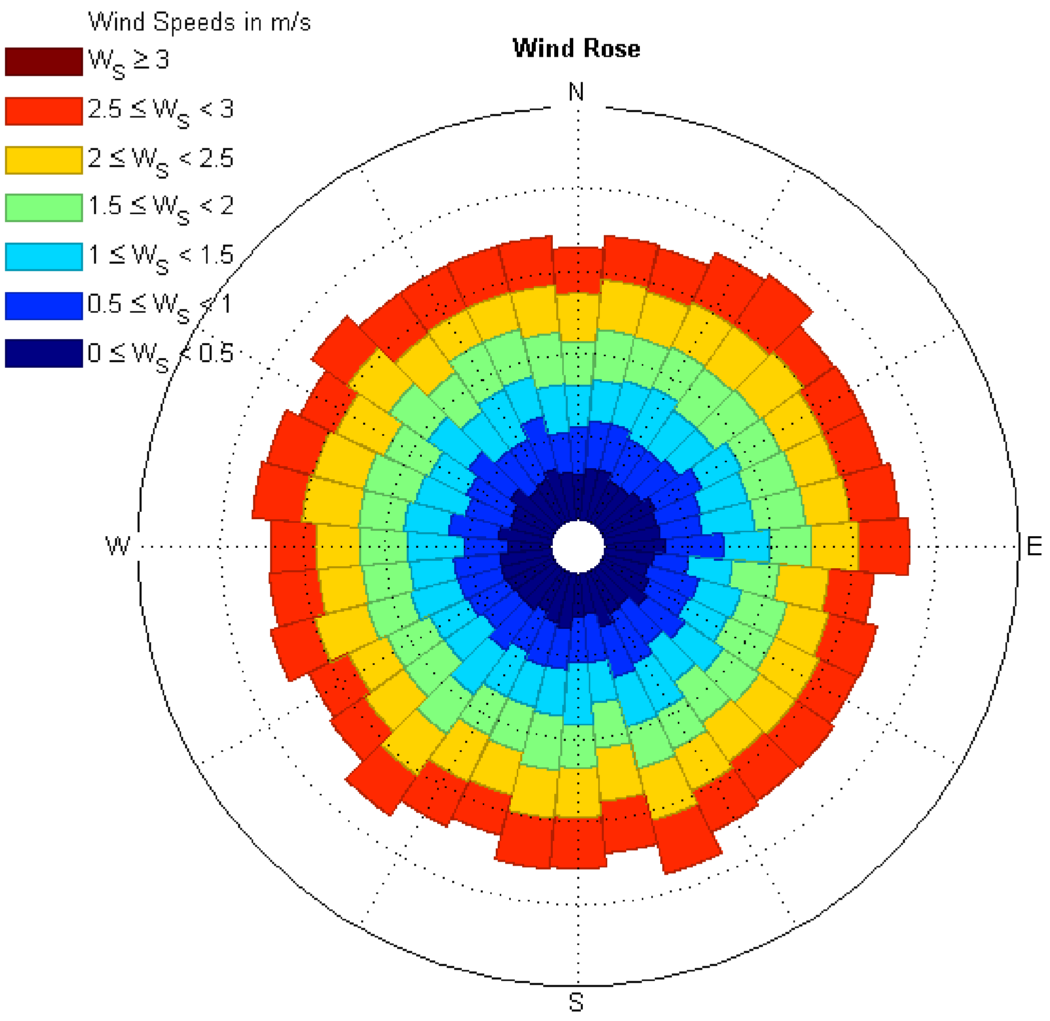


The default option represents 36 directions (each 10º) Note that the bins are centered in 0° (-5° to 5°). We will show how to change that.

Reference angles

As there will always be controversy about which should be the reference angles, this function uses by default the trigonometric convention: Counterclockwise, with 0° angle in the right of the circle. If you want to define your own convention, use sexagesimal degrees to define which angle corresponds to North (up) and to East (right), so any user can use the desired references. Let's imagine that our data uses the noon solar convention (0° is South, and 90° is West - thus we have origin and orientation -) => North is 180° and East is 270°. These two values must differ in 90° and both values needs to be specified.

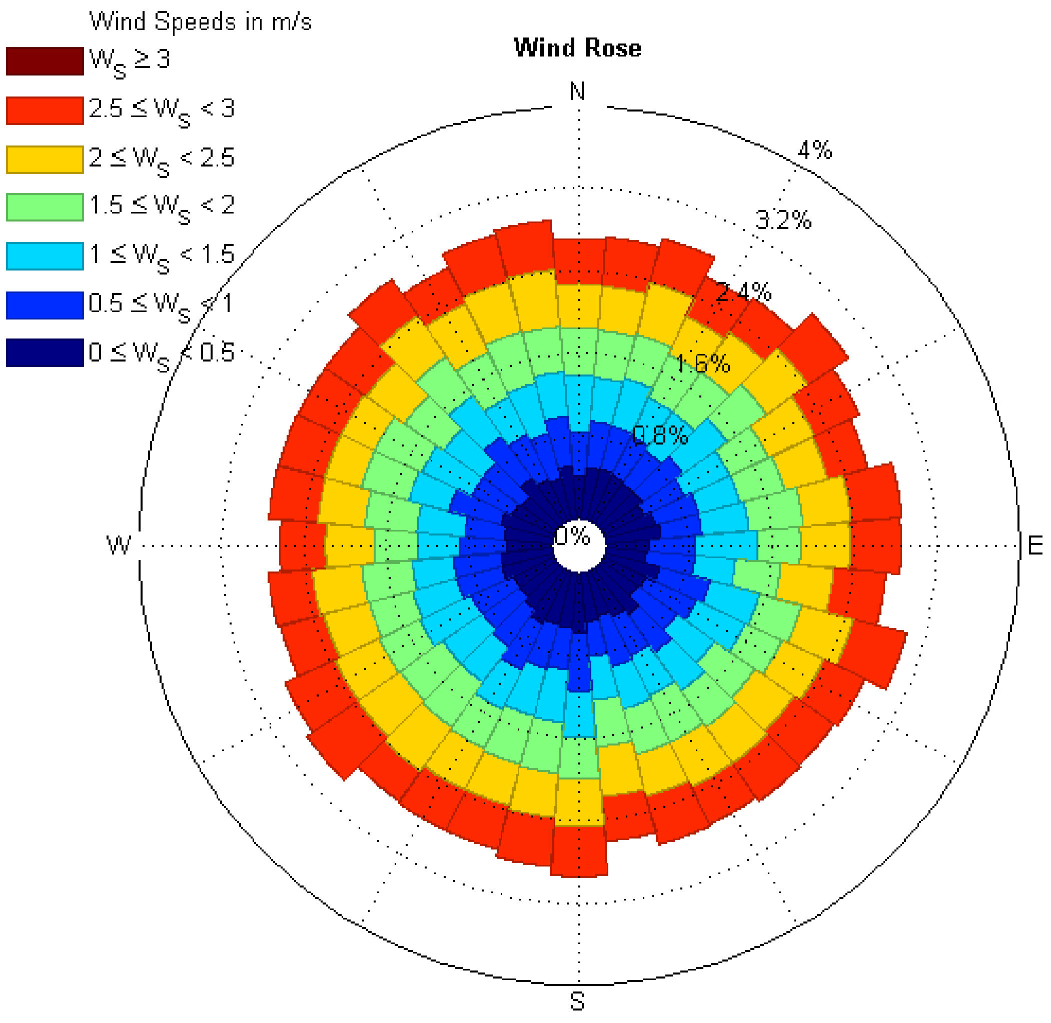
```
[figure_handle,count,speeds,directions,Table] =
WindRose(dir,spd,'AngleNorth',180,'AngleEast',270);
```



Adding the frequency labels

If we want to know the frequency in each direction, it is recommended to add the frequency labels, in any angle

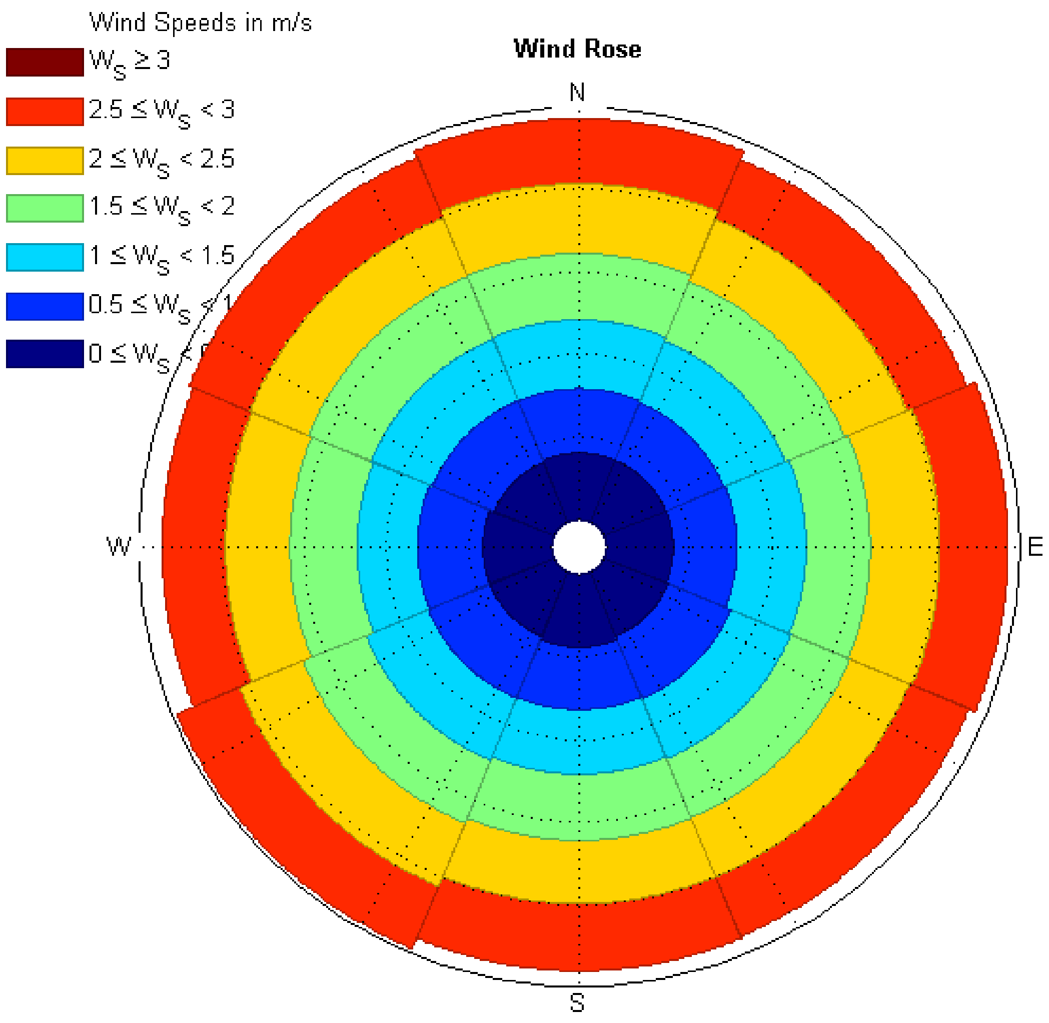
```
[figure_handle,count,speeds,directions,Table] = WindRose(dir,spd,'FreqLabelAngle',60);
```



Changing the number of directions

The number of directions can be changed by adding the number of directions we want to show, let's say, 8 (N, NE, E, SE, S, SW, W and NW)

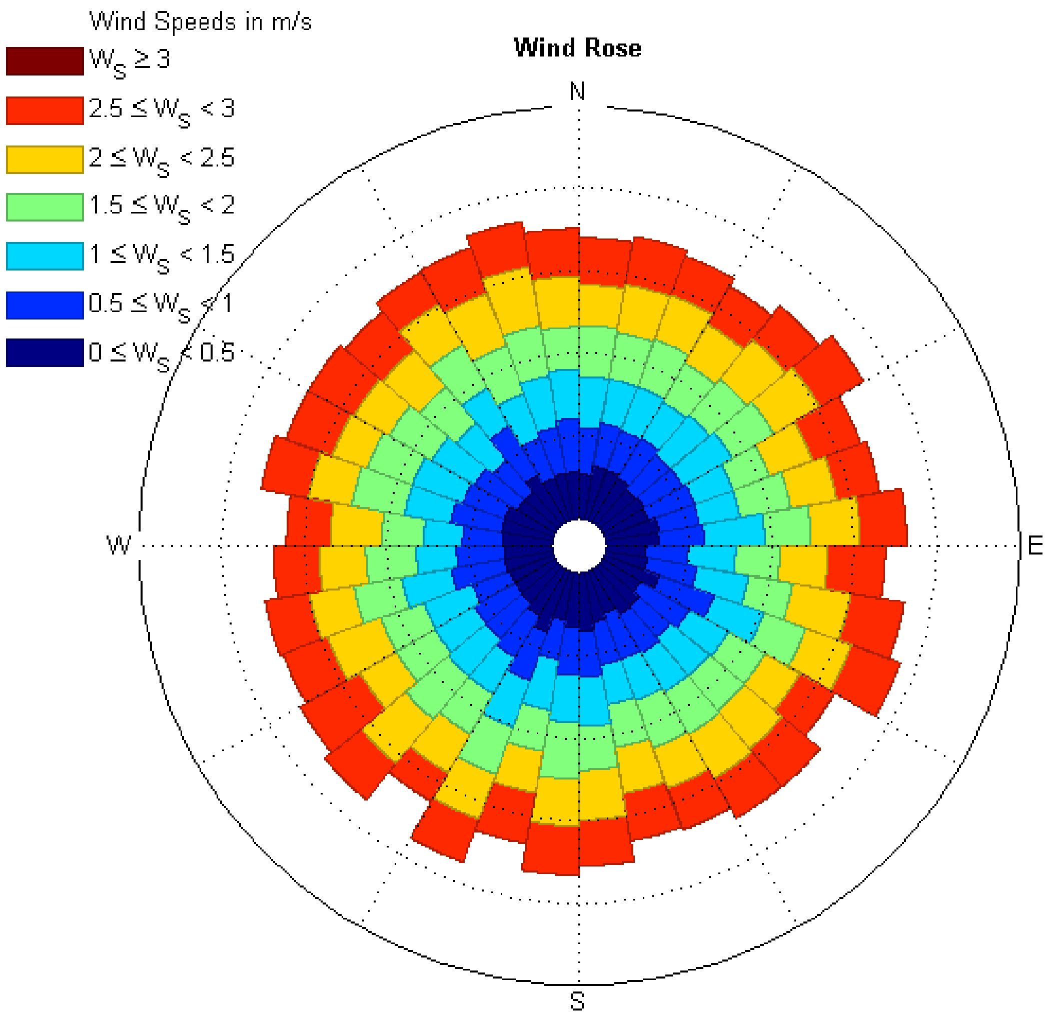
```
[figure_handle,count,speeds,directions,Table] = WindRose(dir,spd,'nDirections',8);
```



Bin alignment

If you do not want the bins to be centered in 0° direction, but you want them starting at that point, you can specify that bins should not be centered in 0:

```
[figure_handle,count,speeds,directions,Table] = WindRose(dir,spd,'CenteredIn0',false);
```

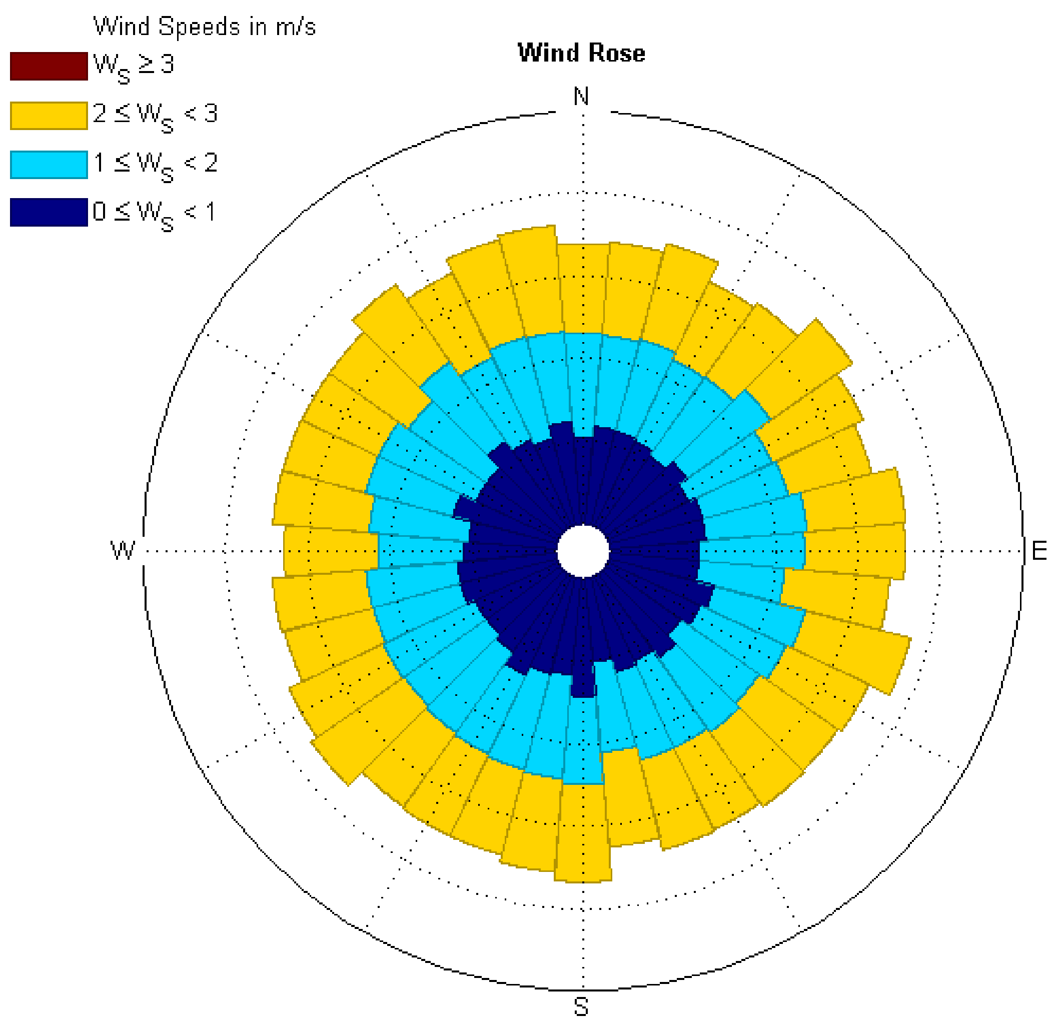


Compare the bins to the first example, now bins appear in the direction 0° to 10°, centered in 5°. The wind rose computes the number of directions pointing in the current range, so the figure is not just rotated with respect to the original, but the values are calculated again.

Changing the number of intensities

Not only can the direction bins be modified. The number of bins for the intensities can be also changed:

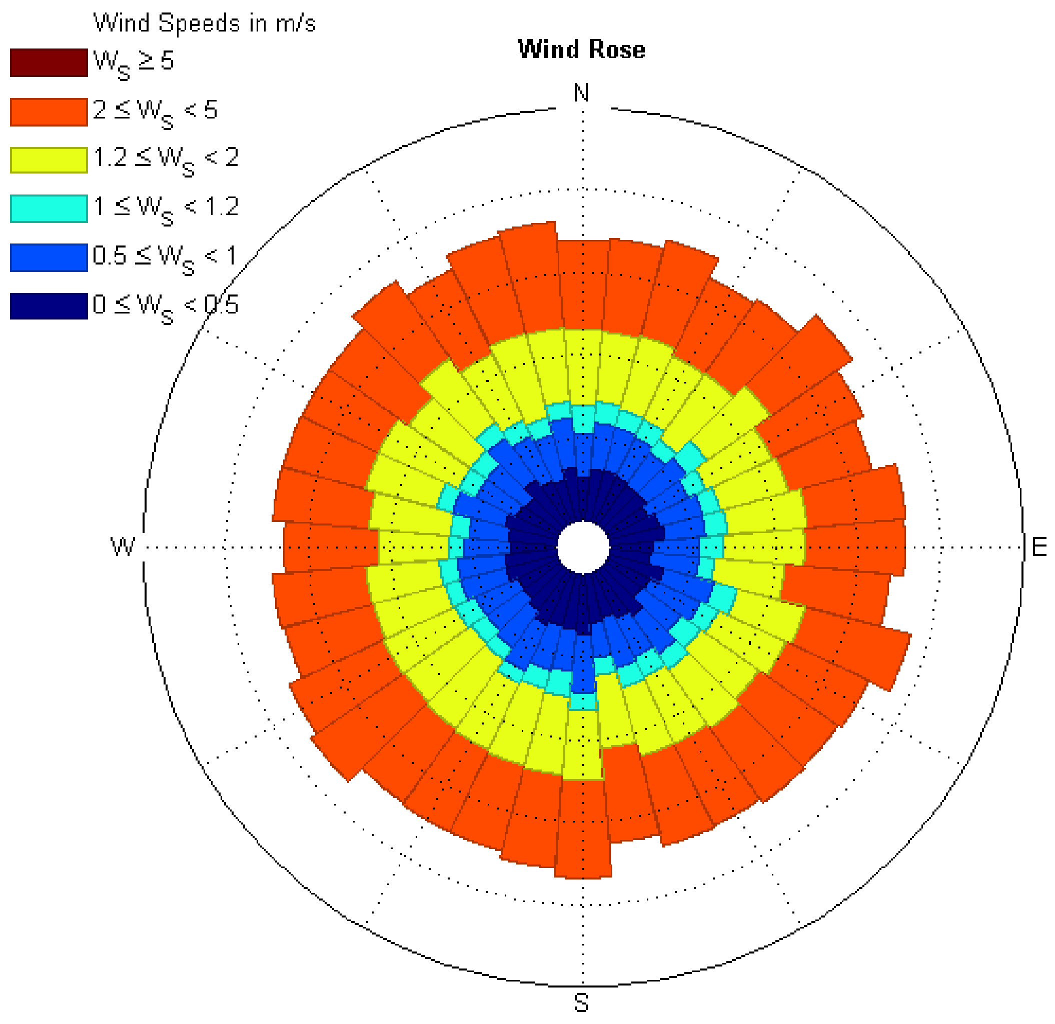
```
[figure_handle,count,speeds,directions,Table] = WindRose(dir,spd,'nSpeeds',4);
```



Defining the intensity ranges

If you prefer defining the speeds to create the bins, you can also do that (this will omit the 'nspeeds' command - use whichever you need). Beware of this, since you have to know the speed ranges, in order to prevent possible errors.

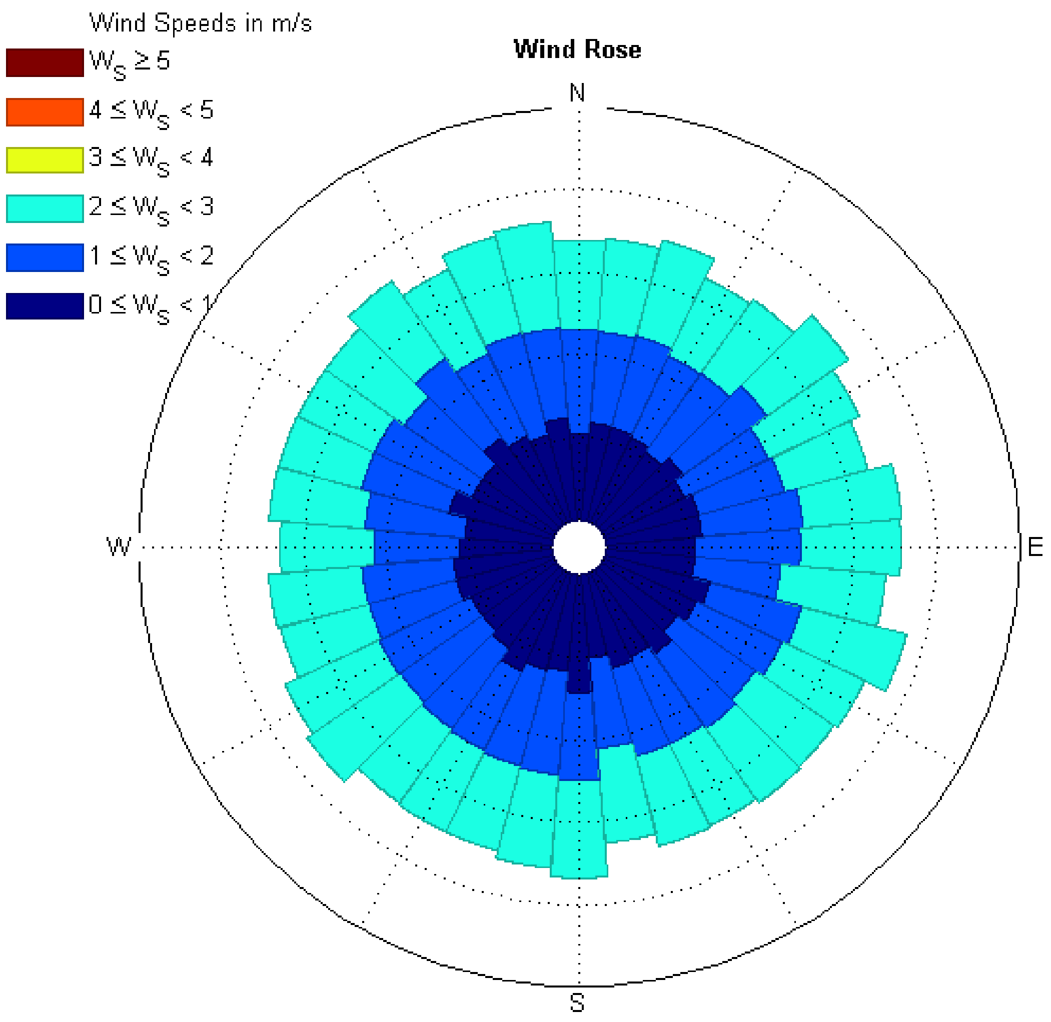
```
[figure_handle,count,speeds,directions,Table] = WindRose(dir,spd,'vWinds',[0 0.5 1 1.2 2
5]);
```



Rounding the maximum intensity

If the maximum intensity is not defined as you wanted, change this (let's imagine that you wanted the maximum speed to be multiple of 5, with 6 intensity ranges, so as all the speed ranges have integer values)

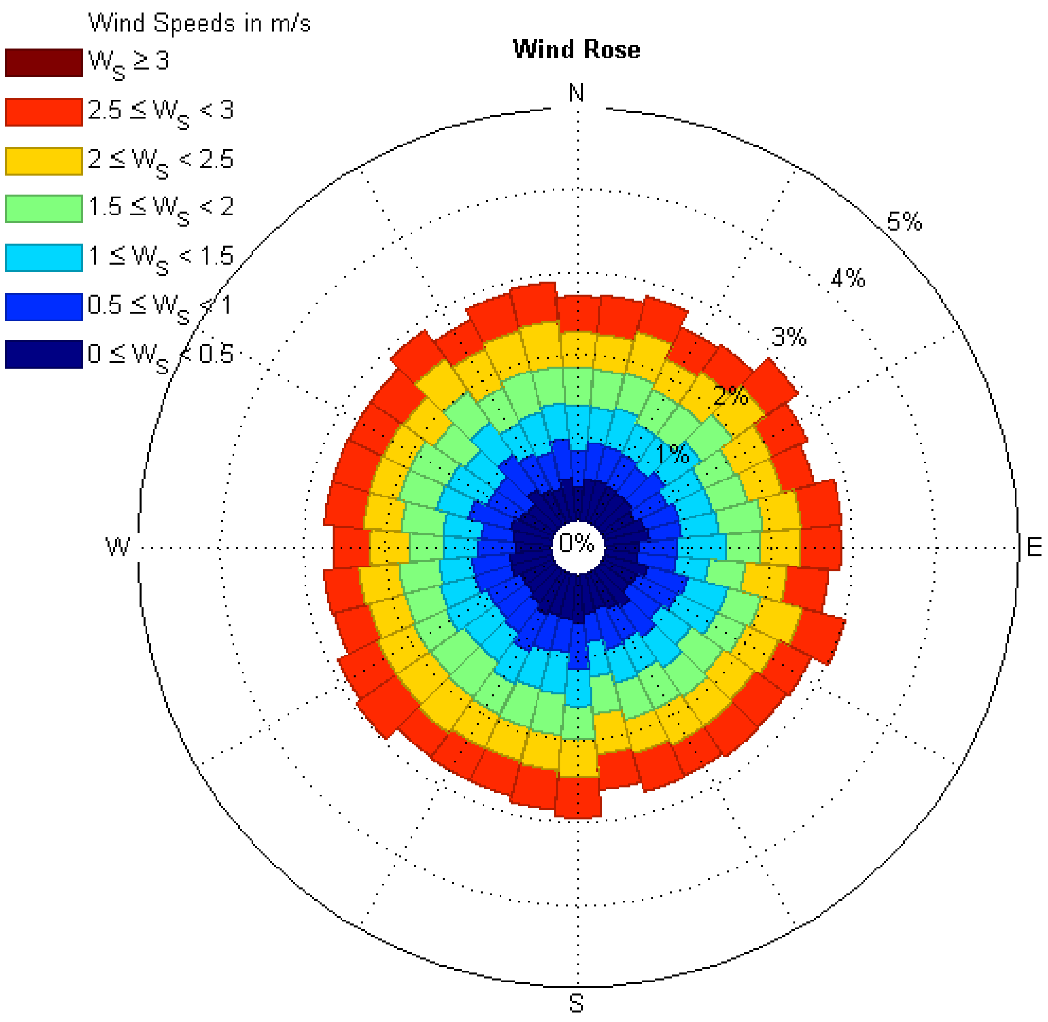
```
[figure_handle,count,speeds,directions,Table] =
WindRose(dir,spd,'SpeedRound',5,'nSpeeds',6);
```



Choosing the maximum frequency to appear

The bins are drawn in the way that they fit in the maximum circle, but, in order to compare different wind roses, it could be interesting to keep a fixed value for the maximum frequency (in percentage, 0-100). Let's add the frequency labels too, in the 45° angle, so we can see the frequencies.

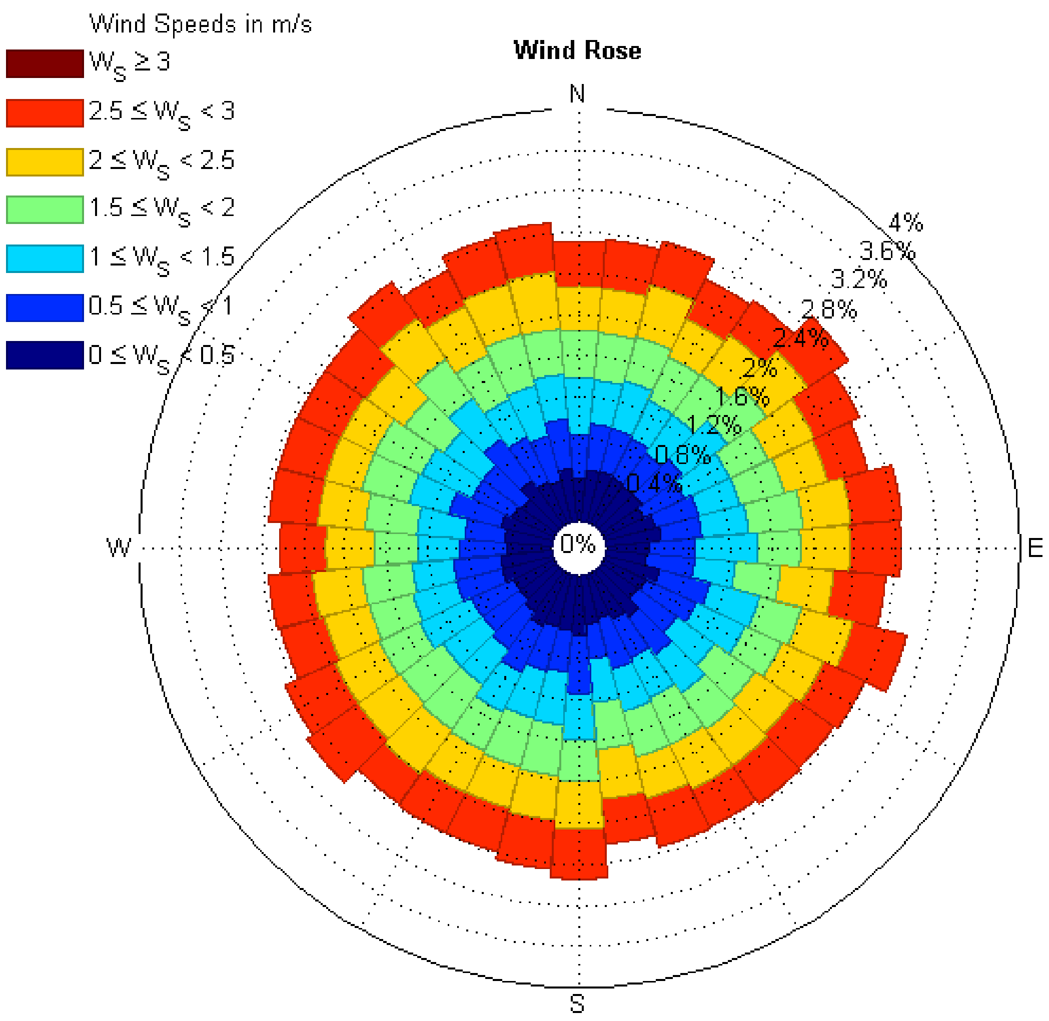
```
[figure_handle,count,speeds,directions,Table] =
WindRose(dir,spd,'MaxFrequency',5,'FreqLabelAngle',45);
```



Changing the frequency lines

In order to check the frequencies, it can be hard if data are very scattered. We can add as many frequency "circles" as we want. Let's add the labels in the 45° line too.

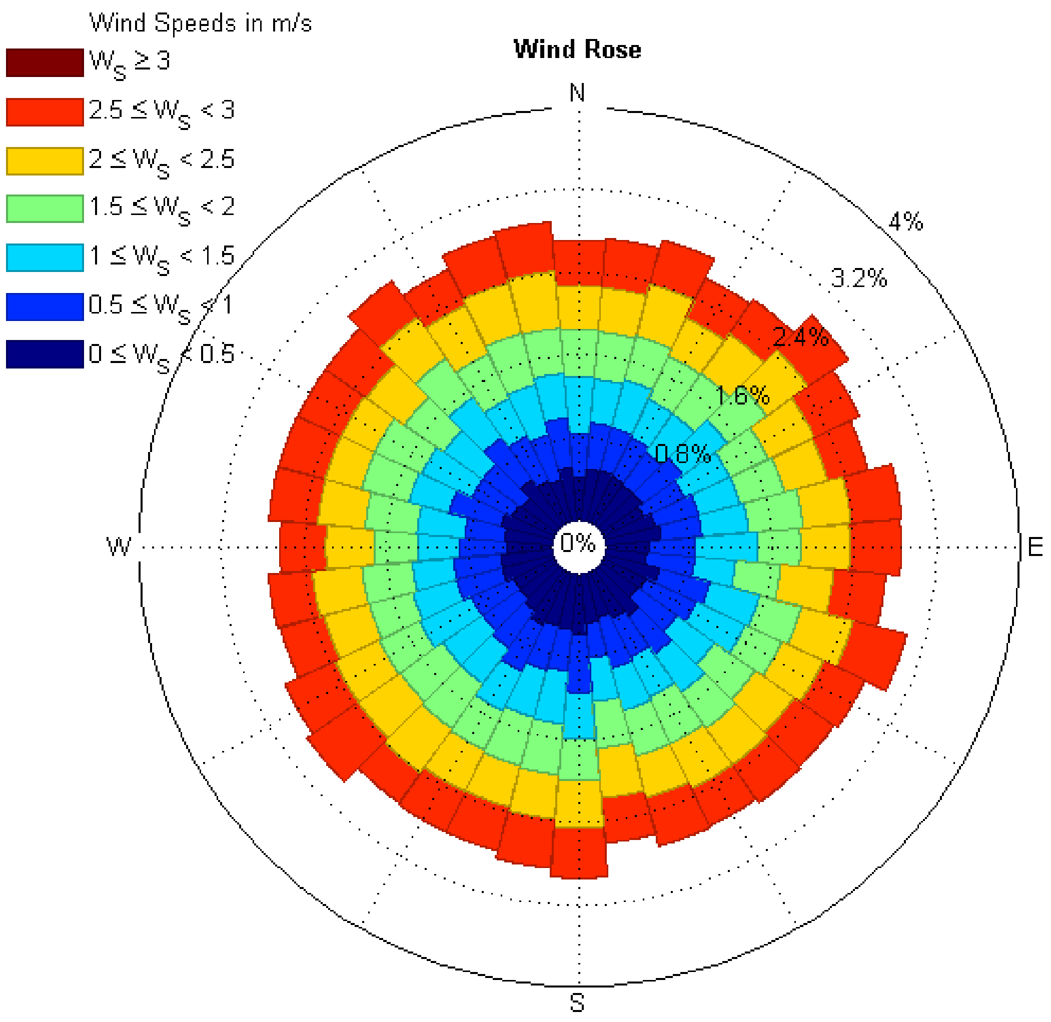
```
[figure_handle,count,speeds,directions,Table] =
WindRose(dir,spd,'nFreq',10,'FreqLabelAngle',45);
```



Rounding the maximum frequency

As we did with the maximum intensity, rounding the maximum frequency can be also interesting, in order to compare very different wind roses. Frequency labels are shown, in order to understand this effect.

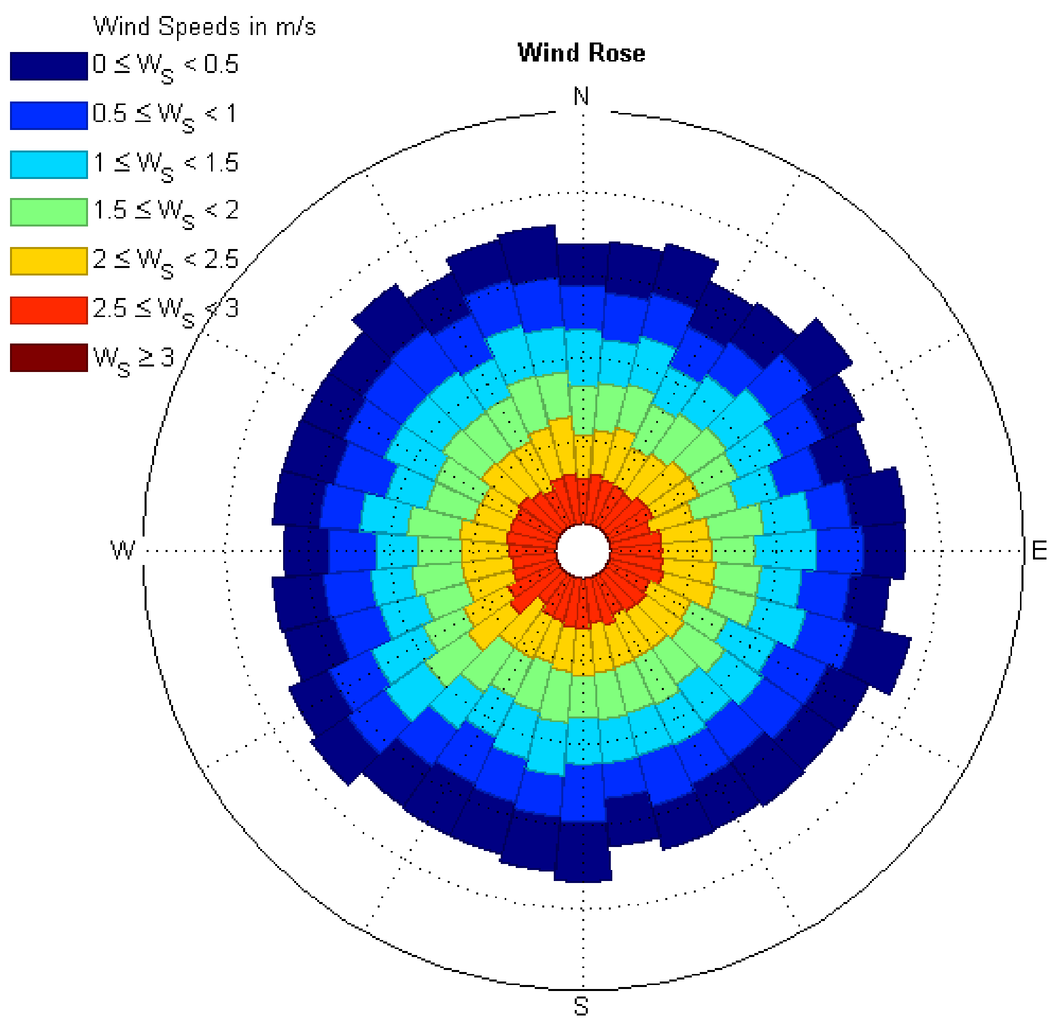
```
[figure_handle,count,speeds,directions,Table] =
WindRose(dir,spd,'FreqRound',2,'FreqLabelAngle',45);
```



Lowest speeds at the outermost bin

If you want to show the lowest speeds at the outermost part of the wind rose, there is a possibility to do this:

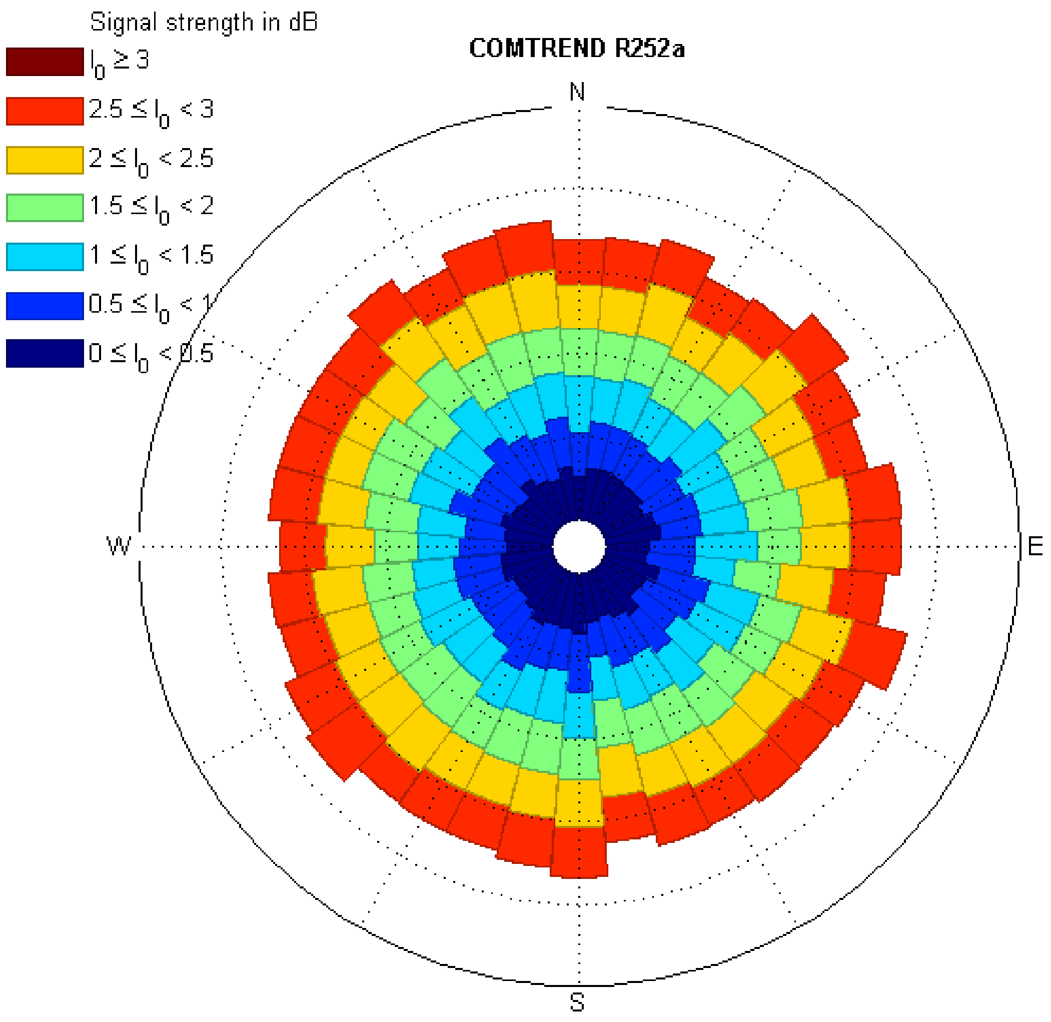
```
[figure_handle,count,speeds,directions,Table] = WindRose(dir,spd,'inverse',true);
```



Changing the title of the figure, and the variable appearing in the legend

Are you using WindRose for other representation that does nothing to do with wind? Change the title the legend's label and the variable. You can use TeX strings too.

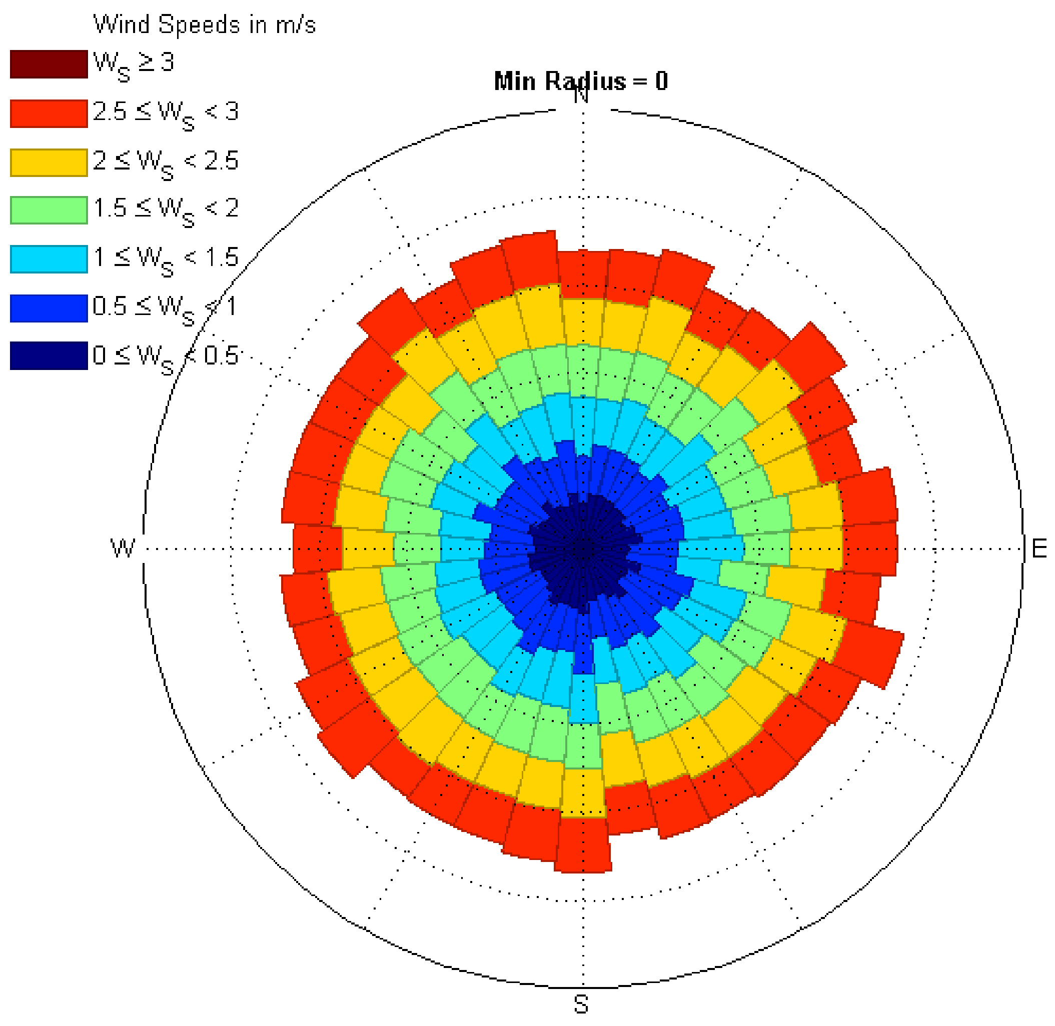
```
[figure_handle,count,speeds,directions,Table] =
WindRose(dir,spd,'TitleString',{'COMTREND R252a';''},'LabLegend','Signal strength in
dB','LegendVariable','I_0');
```

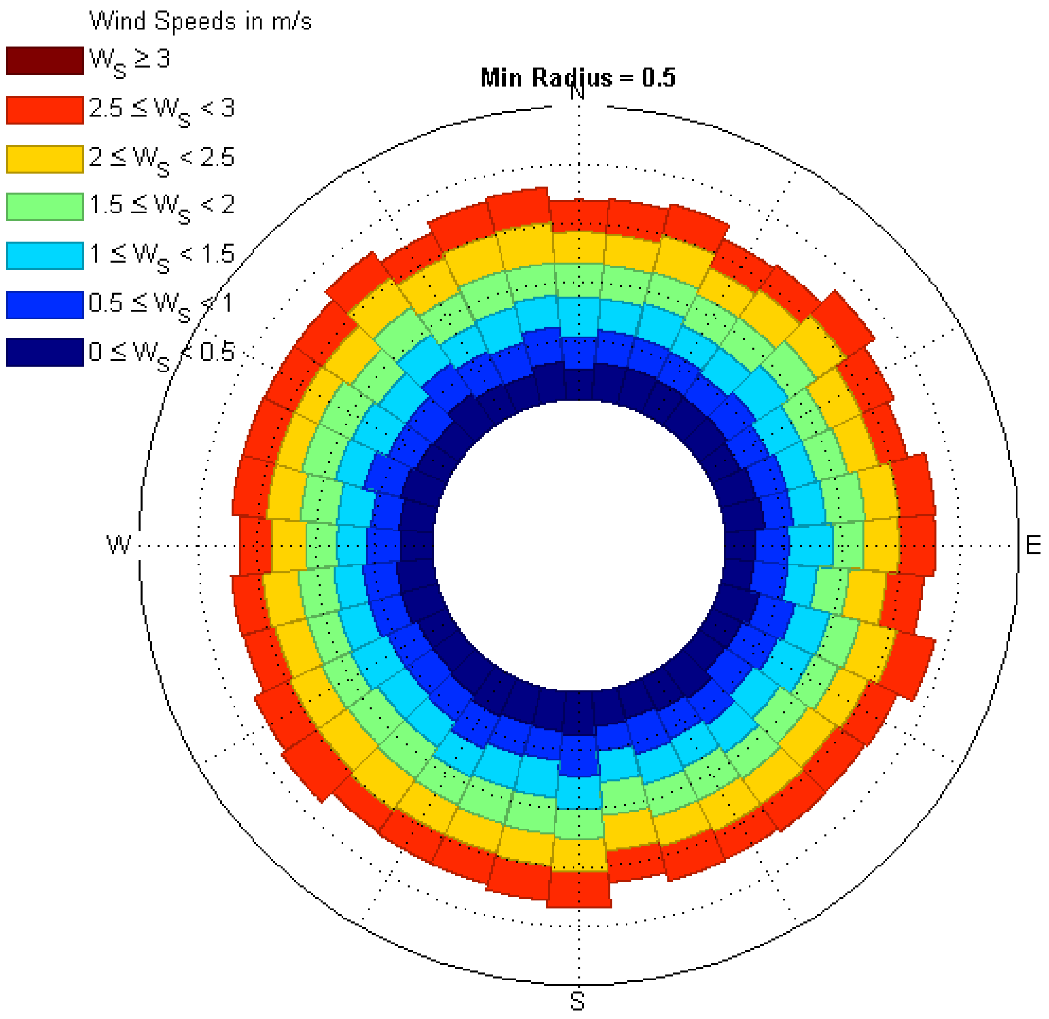


Minimum bin radius

Some users do not like wind roses with a hole in the middle, but others like it a bit wider. Change this attribute by changing the min_radius value, which is relative to the circle radius (p.u., between 0 and 1). The default value is 1/15.

```
[figure_handle,count,speeds,directions,Table] =
WindRose(dir,spd,'Min_Radius',0,'TitleString','Min Radius = 0');
[figure_handle,count,speeds,directions,Table] =
WindRose(dir,spd,'min_radius',0.5,'TitleString','Min Radius = 0.5');
```

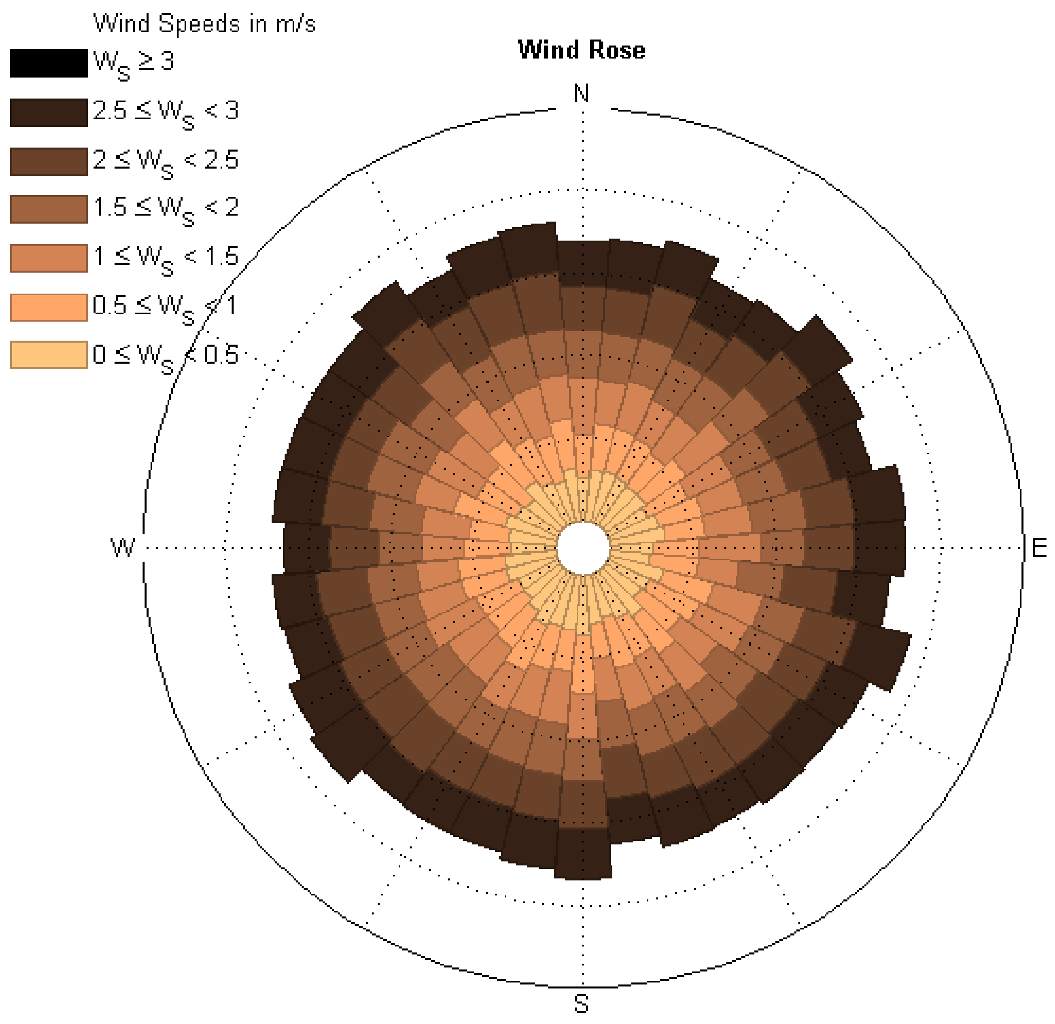



Wind Speeds in m/s
$W_S \geq 3$
$2.5 \leq W_S < 3$
$2 \leq W_S < 2.5$
$1.5 \leq W_S < 2$
$1 \leq W_S < 1.5$
$0.5 \leq W_S < 1$
$0 \leq W_S < 0.5$
Min Radius = 0.5
N
E
S
W

Changing the colormap

If you hate the "jet" colormap -default-, you can use any other built in colormap you want. You can also invert the colormap just by adding "inv" at the beginning of the colormap name. In the example, we will use the "copper" colormap but inverted:

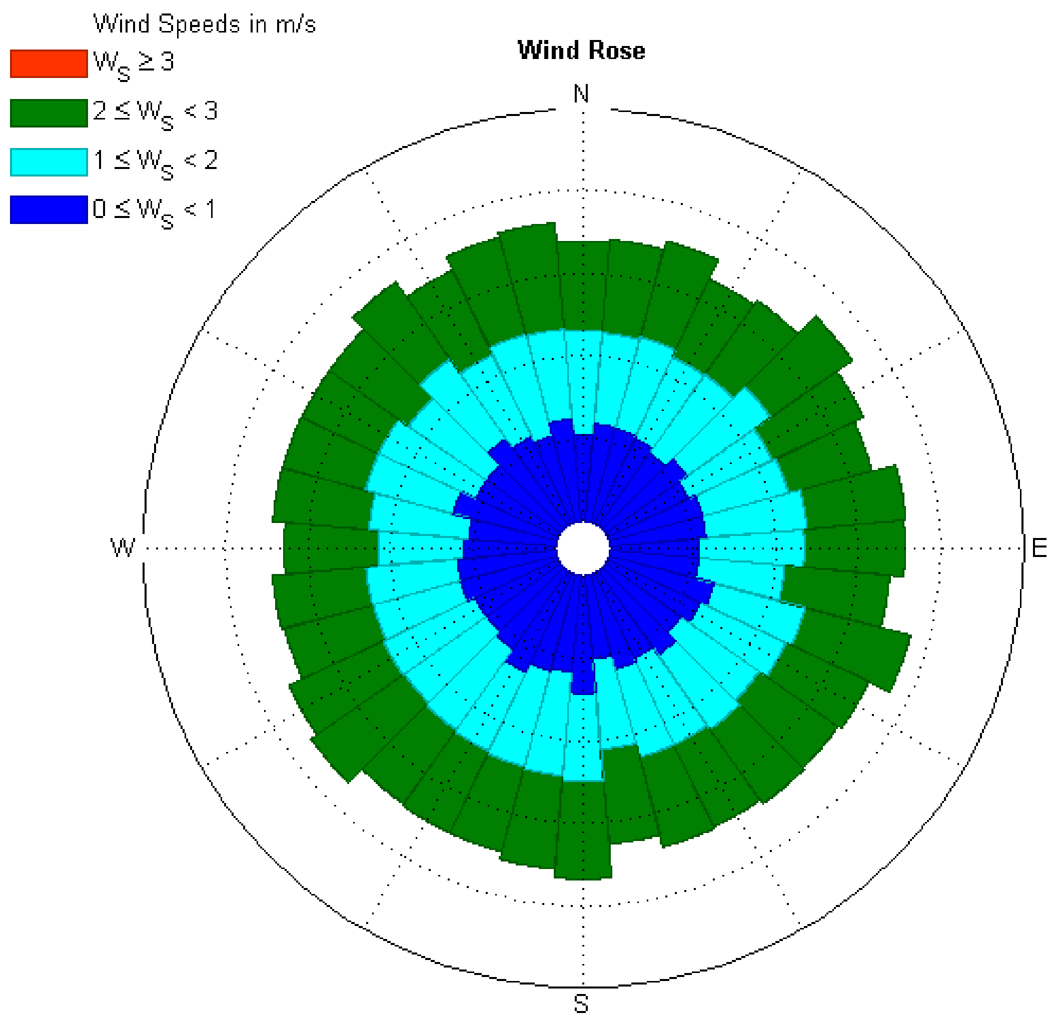
```
[figure_handle,count,speeds,directions,Table] = WindRose(dir,spd,'cMap','invcopper');
```



Custom colormap

If you specify the 'nSpeeds' argument to define the number of ranges, a customized colormap can be used, specifying a numeric array:

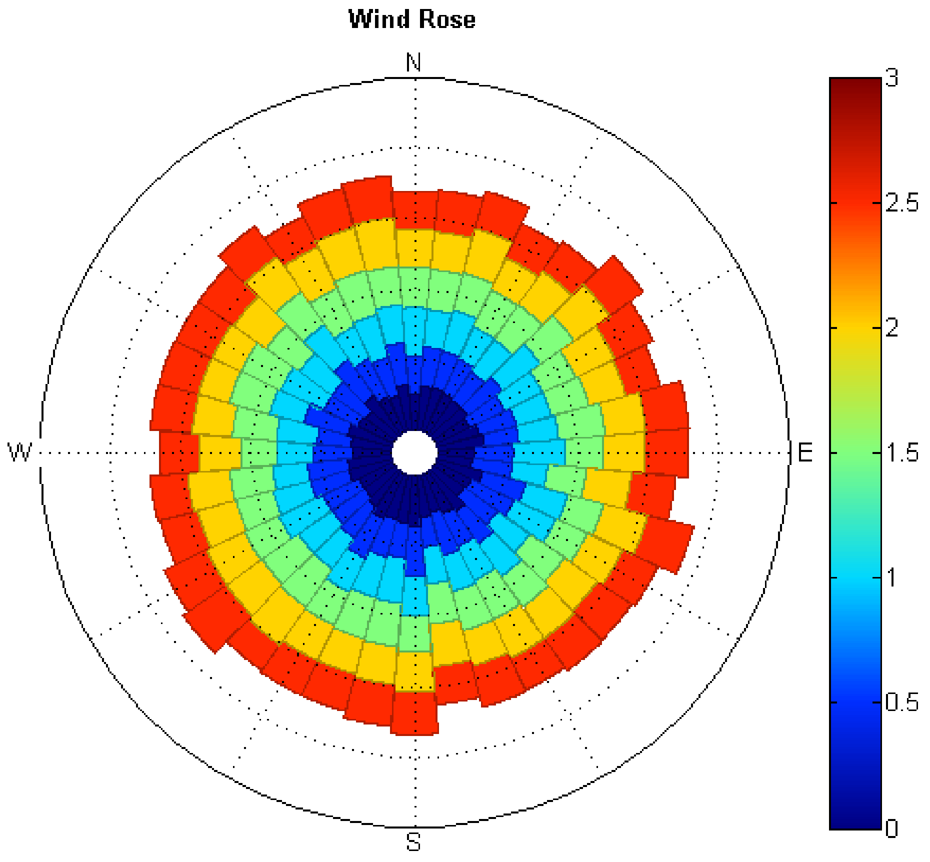
```
[figure_handle,count,speeds,directions,Table] = WindRose(dir,spd,'nSpeeds',4,'colors',[0
0 1; 0 1 1; 0 0.5 0; 1 0.2 0]);
```



Changing the legend

The legend can be two types: Boxes legend -default- (type 2) or colorbar legend (type 1). If you do not want the legend to appear, consider there is an extra type (type 0):

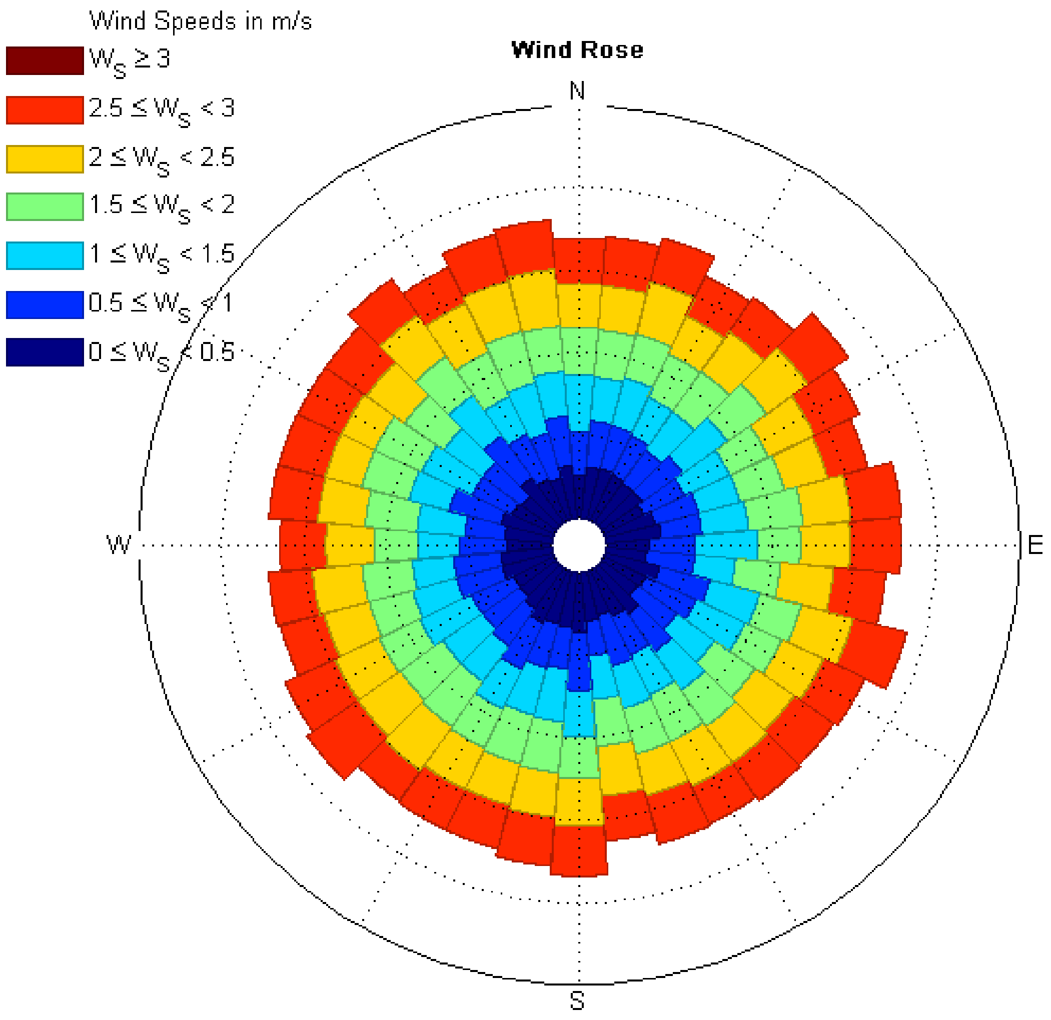
```
[figure_handle,count,speeds,directions,Table] = WindRose(dir,spd,'LegendType',1);
```



Title font weight

If you want the title to appear with a different weight (other than bold) you can choose between normal, light, demi and bold:

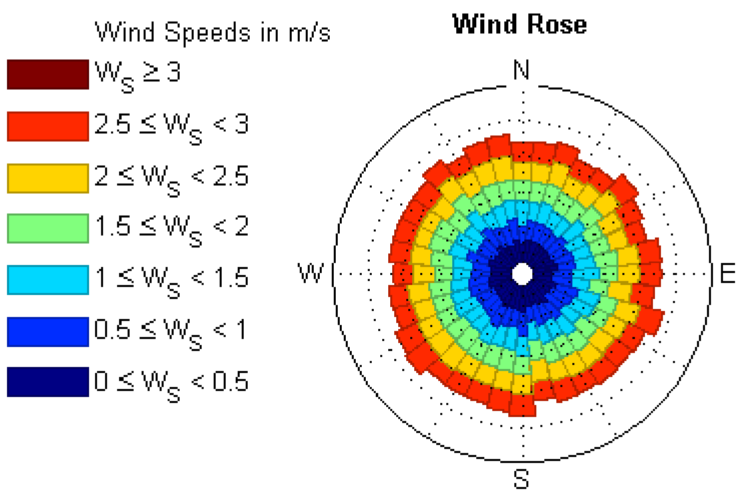
```
[figure_handle,count,speeds,directions,Table] =
WindRose(dir,spd,'TitleFontWeight','demi');
```



Modifying the figure dimensions

The default figure will be squared, with the length of the square being 2/3 of the smallest dimension of the screen (usually the height). You can specify the figure dimensions in pixels. These dimensions will be the inner figure dimensions (window border is extra)

```
[figure_handle,count,speeds,directions,Table] =
WindRose(dir,spd,'height',256,'width',512);
```



Modifying the scale of the wind rose inside the figure

If you consider that the wind rose is too big inside the figure, you can reduce its size by using an scale factor (0 to 1)

```
[figure_handle,count,speeds,directions,Table] = WindRose(dir,spd,'scalefactor',0.5);
```

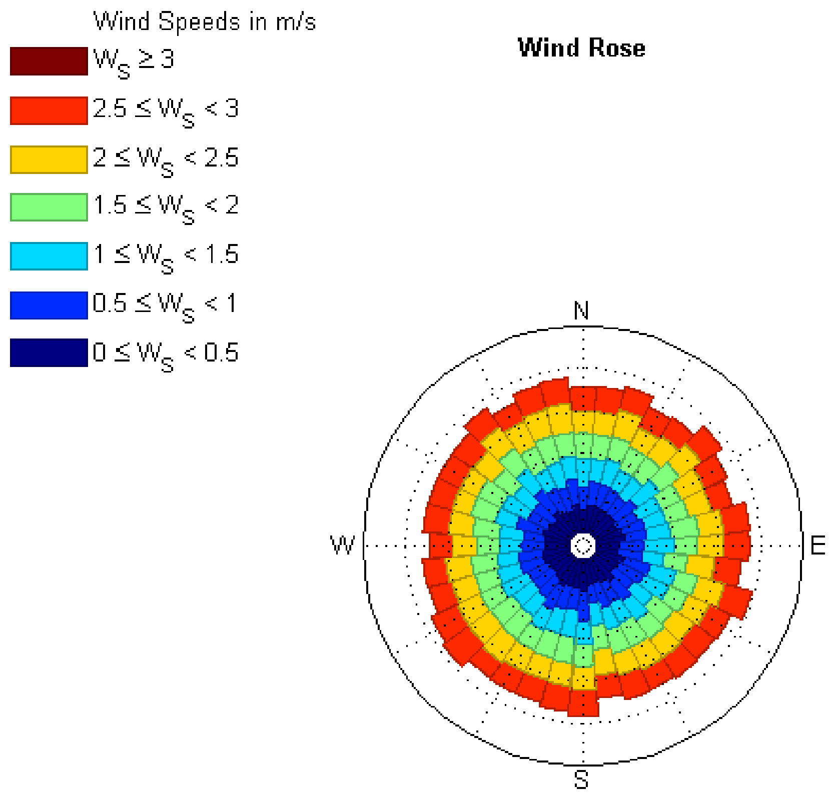


Figure toolbars

If you want to remove the figure and menu bars, you can use the same instructions used in a normal figure:

```
[figure_handle,count,speeds,directions,Table] =
WindRose(dir,spd,'menubar','none','toolbar','none');
```

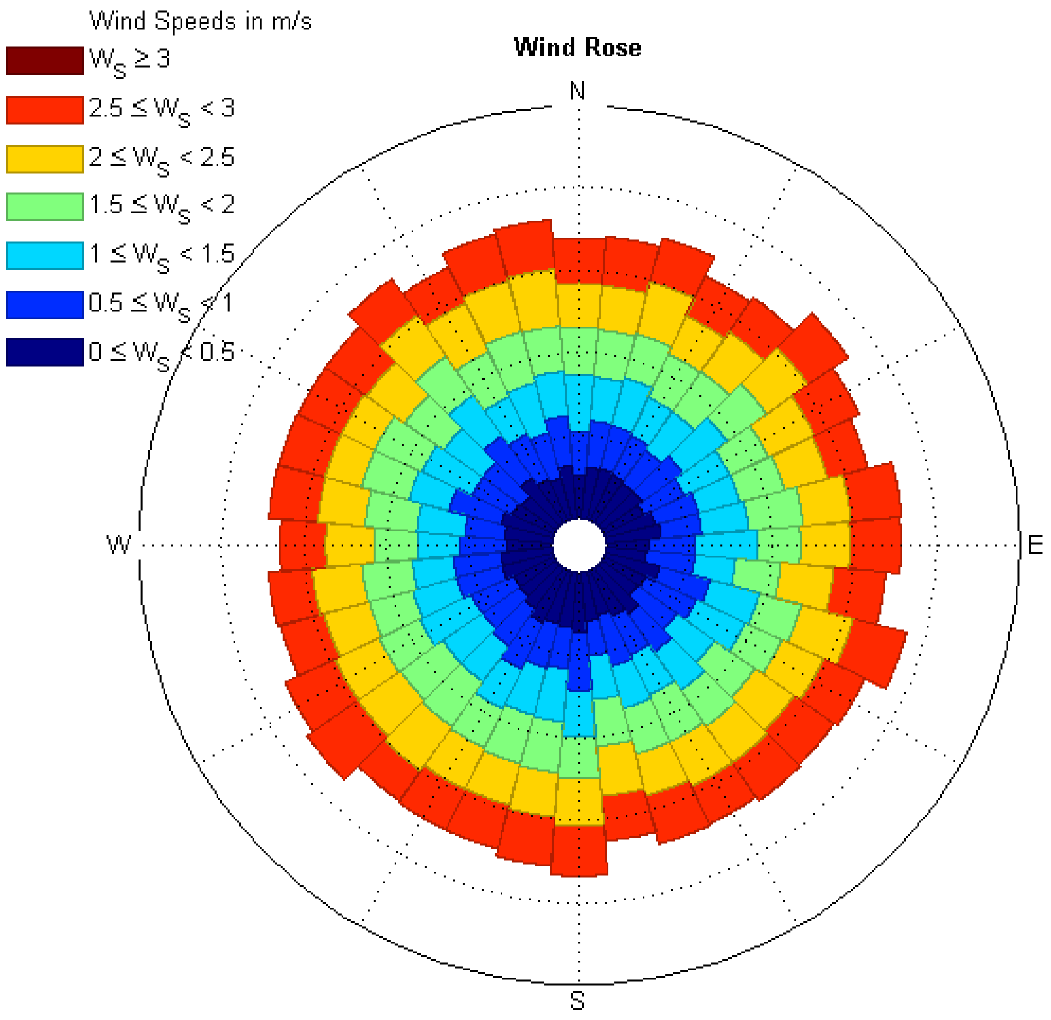
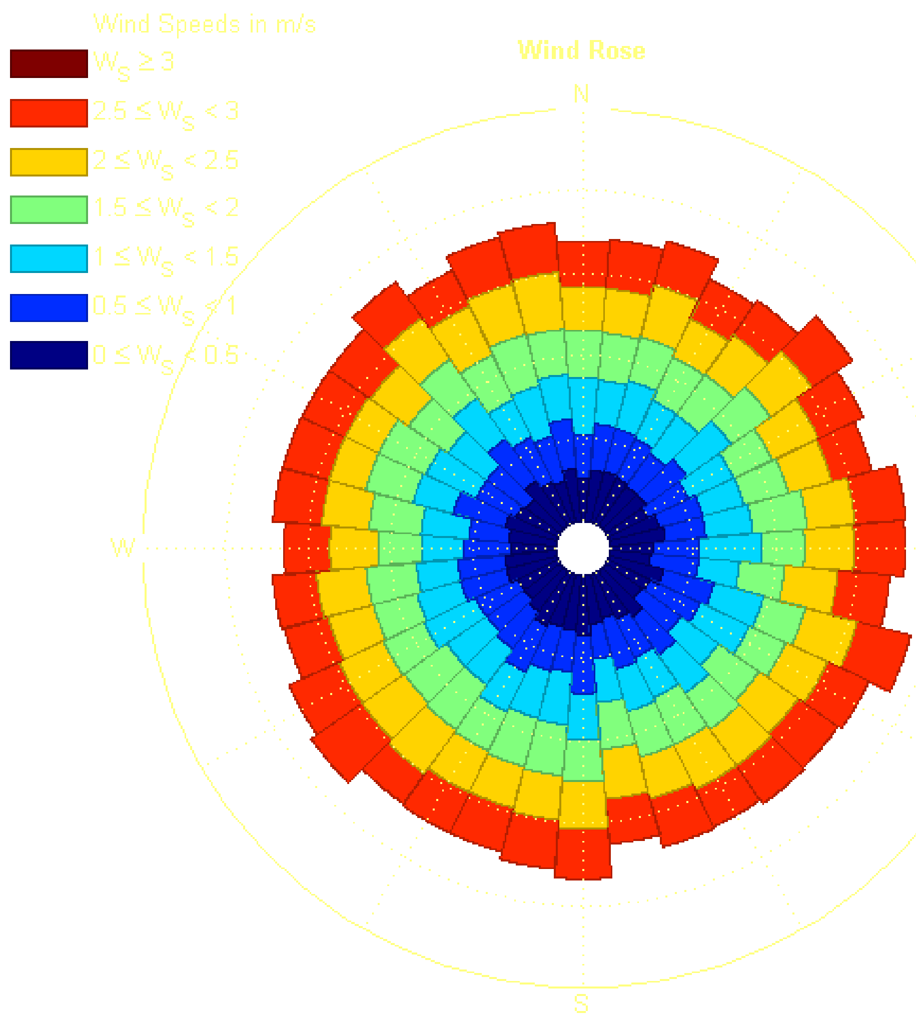


Figure and text colors

Figure is white by default; while text appears to be black. Modify them as wanted. Use Matlab built-in colors or use custom RGB vector:

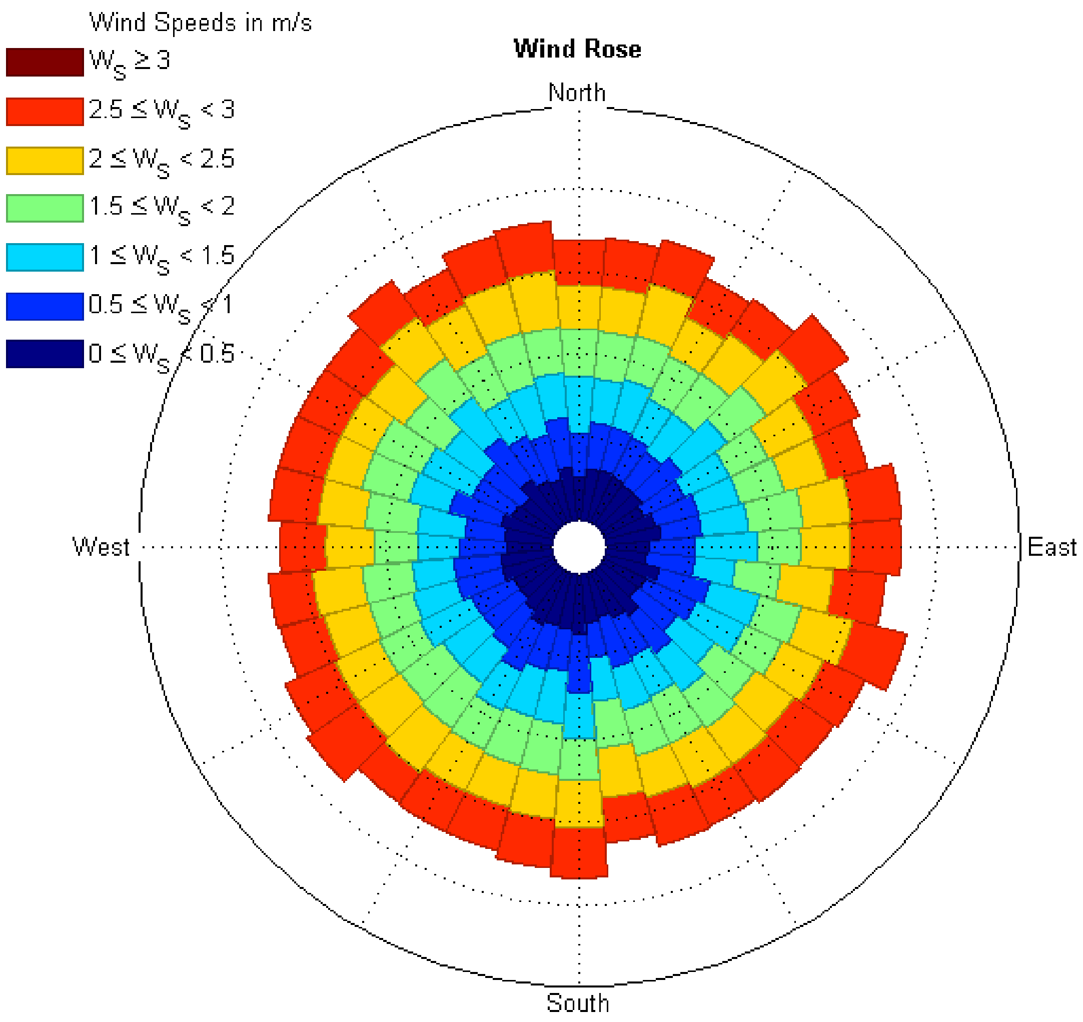
```
[figure_handle,count,speeds,directions,Table] =
WindRose(dir,spd,'figColor','k','textColor',[1 1 0.5]);
```

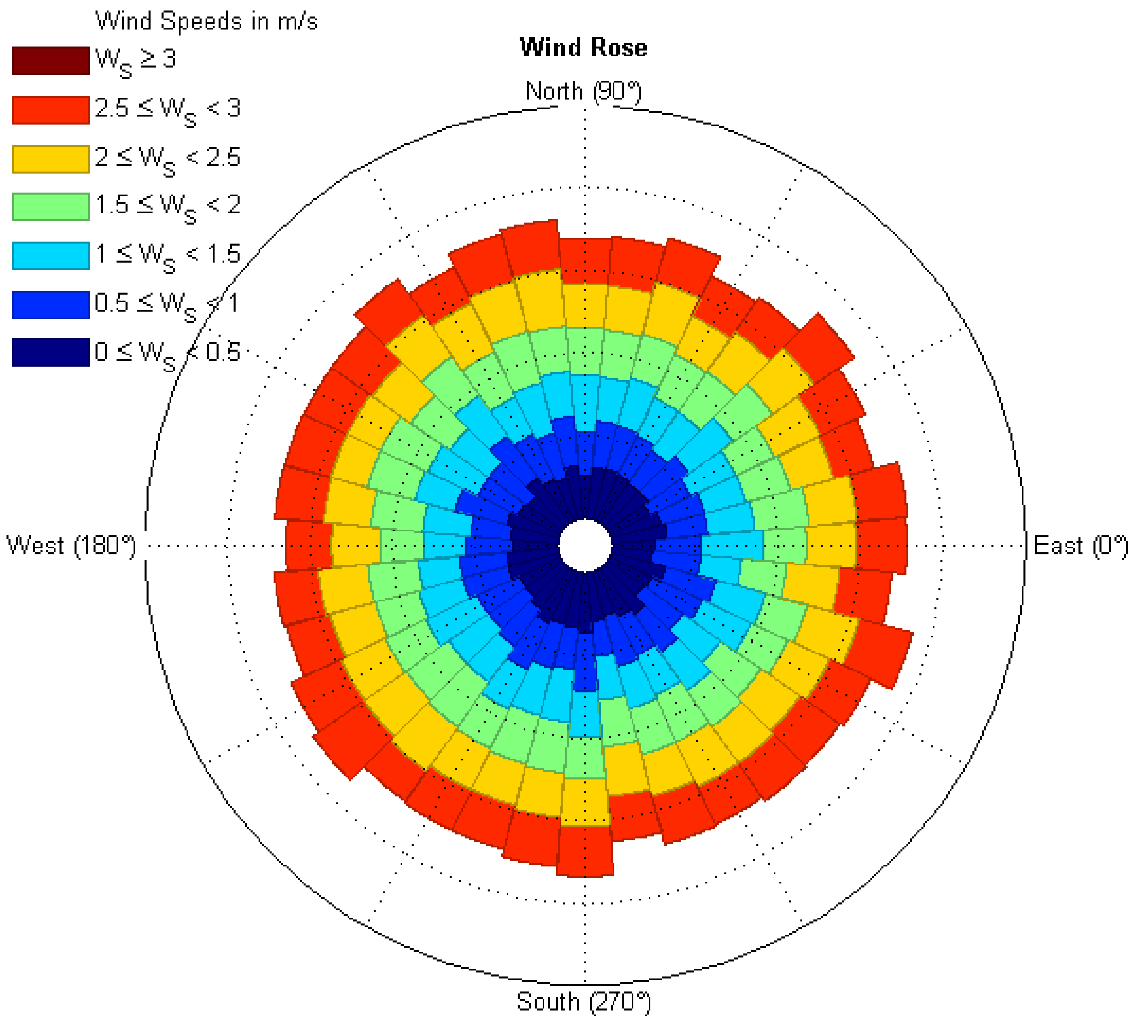


Axis labels

Axis labels -default are N, S, E and W- can be modified with separate properties or with a cell array (North, South, East and West):

```
[figure_handle,count,speeds,directions,Table] =
WindRose(dir,spd,'labelNorth','North','labelSouth','South','labelEast','East','labelWest
','West');
[figure_handle,count,speeds,directions,Table] = WindRose(dir,spd,'labels',{'North
(90°)','South (270°)','East (0°)','West (180°)'});
```



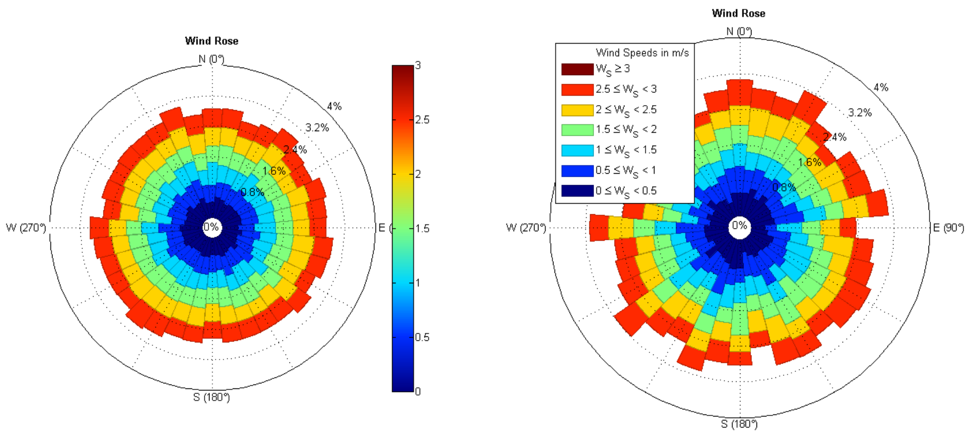

Wind Speeds in m/s
$W_S \geq 3$
$2.5 \leq W_S < 3$
$2 \leq W_S < 2.5$
$1.5 \leq W_S < 2$
$1 \leq W_S < 1.5$
$0.5 \leq W_S < 1$
$0 \leq W_S < 0.5$
Wind Rose
North (90°)
East (0°)
South (270°)
West (180°)

Creating different wind roses in different axes (subplot)

To create wind roses in different axes use the 'axes' argument, followed by the handle of the axes in which to plot the wind roses:

```
dir2  = mod(360*rand(5000,1),360);
spd2  = abs(rand(5000,1))*3;
axes1 = subplot(1,2,1);
axes2 = subplot(1,2,2);
set(gcf,'units','normalized','position',[0 0 1 1]);
Options1 = {'anglenorth',0,'angleeast',90,'labels',{'N (0°)','S (180°)','E (90°)','W (270°)'},'freqlabelangle',45,'axes',axes1,'legendtype',1};
[figure_handle,count1,speeds1,directions1,Table1] = WindRose(dir,spd,Options1);

Options2 = {'anglenorth',0,'angleeast',90,'labels',{'N (0°)','S (180°)','E (90°)','W (270°)'},'freqlabelangle',45,'axes',axes2};
[figure_handle,count2,speeds2,directions2,Table2] = WindRose(dir2,spd2,Options2);
```



Outputs

This function has several outputs.

4. **figure_handle**. Type *DOUBLE*. Dimensions **1×1**. The first one is the figure handle, which can be used to modify its position, printing, image saving, etc...
5. **count**. Type *DOUBLE*. Dimensions **d×s**. count is a two dimensional array, where the frequency for each intensity (in columns) and direction (in rows) is represented.
6. **speeds**. Type *DOUBLE*. Dimensions **1×s**. speeds returns a 1-D array with the speeds appearing in the legends. The number of elements in this array matches the number of columns in count
7. **directions**. Type *DOUBLE*. Dimensions **d×1**. directions returns a 1-D array with the mean value of the directions of each "branch" in the wind rose. The number of elements in this array matches the number of rows in count
8. **Table**. Type *CELL*. Dimensions **4+d × 3+s**. Table is a summary table where the frequencies for each direction and each speed are shown, in an excel ready format, so this can be used for *xlswrite* or just prompting in Matlab's command window.

Importing data from excel and exporting back to Excel

Within the zip including this function, an extra script (*ExcelImportExport.m*) is included.

This script works only with Windows versions of Matlab + Excel and does the following:

1) Reads a previously created Excel file (*wind data.xlsx*) with wind direction and speed time series.
2) Plots the wind rose and saves it into PNG file.
3) Exports the table result into a new Excel file (*wind data outputs.xlsx*) and
4) Inserts the image inside the new Excel file (*wind data outputs.xlsx*)

Take a look at this script: the code is heavily commented and can be changed in order to import/export from/to whatever file, sheet or range in Excel.

Exporting the figure to a PNG file

To export the figure into a PNG file, you can use the command

```
print('-dpng', 'WindRose.png','-painters');
```

Or you can use any of the numerous contributions in the Mathworks File Exchange page, from which I recommend *export_fig*, by Yair Altman (who is responsible for the amazing blog http://www.undocumentedmatlab.com/).

You can find the *export_fig* submission at:

https://www.mathworks.com/matlabcentral/fileexchange/23629-export-fig