Rcost Introduction

Reto Stauffer
University of Innsbruck

October 4, 2017

Abstract

This package contains a user-friendly and handy interface to the COST Action 733 WLK (Wetterlagenklassifikation) algorithm. The motivation of the COST Action 733 was to "achieve a general numerical method for assessing, comparing and classifying weather situations in Europe, scalable to any European (sub)region with time scales between 12 hours and 3 days and spatial scales of ca. 200 to 2 000 km."

Details can be found on `http://cost733.met.no/`. This R package interfaces one routine which was developed during the COST 733 Action with some minor modifications. Not all tuning parameters are implemented yet. Implementation of these parameters should be feasible if needed.

1 COST Action 733: WLK

The WLK (Wetterlagenklassifikation) algorithm is a relatively simple grid-based method to classify large-scale weather patterns. The `Rcost` package interfaces the fortran code `cost733_wlk_1.07.f90` which can be found here: http://cost733class.geo.uni-augsburg.de/moin/cost733wiki/data/pages_bak/WLKC733/

The analysis uses the two wind speed components (`u` zonal, `v` meridional) in 700 hecto Pascal, the (geopotential) height of the 500 hecto Pascal and 925 hecto Pascal surface plus (`z` in our case) the total column water content (`tcw`) of a pre-specified (sub)region.

Thus, the classification can easily be performed on analysis or re-analysis data (as in the package example), but also on numerical weather prediction (NWP) forecasts. Please note that only regular latlong grids are supported!

2 Methodology

Originally the COST 733 WLK classification returns a set of strings and IDs for the prevailing weather situation. The IDs are unique for a specific combination classified geopotential pattern (cyclonic/anticyclonic on 500/850 hPa), main wind direction, and whether the atmosphere is dry or wet given a baseline climatology.

`Rcost` does not return the combined classes. Instead the raw numeric values (plus binary classes) are returned. As an example: the original algorithm classifies cyclonicity as `1` (anticyclonicity as `0`) but looses the information about the strength of the curvature. `Rcost` returns both, binary `0/1`, but also the numeric value for the curvature which is very useful for certain applications.

In general: the weights

For all 'variables' or sub-classes in the WLK classification a spatial weighting scheme is used (based on the weight mask, see Subsec. 4.1).

Wet or Dry

`Rcost` is currently using total column water (`tcw`) from the input files. It describes the amount of water contained in a vertical column trough the entire atmosphere and will be returned as "precipitable water index" (`pwi`; see classifier output: Sec. 6).

$$\mathrm{pwi} = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J}\mathrm{tcw}_{i,j}w_{i,j}}{\sum_{i=1}^{I}\sum_{j=1}^{J}w_{i,j}} \tag{1}$$

... which is simply the weighted mean over all $\mathrm{tcw}_{i,j}w_{i,j}$ over all grid cells where tcw is the total column water from the input file, w our weight mask, and (i, j) all grid cells in our domain (with $i = 1, I$ along longitude and $j = 1, J$ along latitude).

As the amount of water uptake of a certain volume of air depends on temperature the values show a strong seasonal signal. Therefore, the pwi values are compared against a background climatology (which has to be provided, see Sec. 4). If the climatology `pwclim` is set when calling `getClassification(...)` the output column `wet` (see Sec. 6) is set to `1` if $\mathrm{pwi}_{\mathrm{yday}} > \mathrm{pwclim}_{\mathrm{yday}}$ end `0` else. Subscript yday indicates a certain day of the year, or Julian day.

Cyclonic or Anticyclonic

The cyclonicity is computed on the geopotential height, for both, the 500 hecto Pascal and 925 hecto Pascal surface. To be specific: this is evaluated on the height (approx.) using z = geopotential height/10.0 (see original Fortran code).

$$c = \frac{z_{i+1,j} + z_{i-1,j} + z_{i+j+1} + z_{i+j-1} - 4z_{i,j}}{\Delta x \Delta y} \tag{2}$$

... for all $i = 2, I-1$ and $j = 2, J-1$. The boundaries (where $i = 1$, $i = I$, $j = 1$, and $j = J$) are filled up with closest $i = 2$, $i = I-1$, $j = 2$, and $j = J-2$. The edges are filled with the values of the closest diagonal grid point (for $(1, 1)$ $(2, 2)$ is used, for $(1, J)$ $(2, J-1)$ and so far and so on). Δx and Δy represent the grid spacing and are computed as follows:

$$\Delta x = \left[\sin(\mathrm{lat}_i)\sin(\mathrm{lat}_i) + \cos(\mathrm{lat}_i)\cos(\mathrm{lat}_i)\cos(\mathrm{lon}_{j-1} - \mathrm{lon}_{j+1})\right]\frac{R\pi}{180.} \tag{3}$$

$$\Delta y = \left[\sin(\mathrm{lat}_{i-1})\sin(\mathrm{lat}_{i+1}) + \cos(\mathrm{lat}_{i-1})\cos(\mathrm{lat}_{i+1})\right]\frac{R\pi}{180.} \tag{4}$$

... where lon and lat are in degrees radiant and $R = 6371.0087714$ is the earth radius in kilometers.

If this value c (output `c500` and `c925`, see Sec. 6) is bigger than 0.0 they will be classified as cyclonic, else anticyclonic. As for `pwi` the `Rcost` package returns both, the numeric value and a binary classification (`0/1`).

Main Wind Sector

Decides whether there was a 'main wind direction or not' based on weighted grid cell wise wind direction and a threshold (`maindirthreshold` see Sec. 4). The wind direction is the meteorological wind direction with 0 and 360 degrees correspond to North wind (wind from North to South), 90 East, 180 South, and 270 West.

Each grid cell is attributed to one of the N wind sectors (`nsector`, see Sec. 4) where each wind sector covers $\frac{1}{N}th$ of 360 degrees. By default, eight wind sectors are considered (`nsector = 8`) wherefore sector `1` is defined as wind direction $[0, 45)$, sector `2` $[45, 90)$ and so far and so on. Given the weights (see Subsec. 4.1) the frequency for each sector is counted:

$$f_n = \sum_{i=1}^{I} \sum_{j=1}^{J} \mathbf{1}_{i,j|n} w_{i,j} \tag{5}$$

f_n denotes the frequency for sector n for each $n = 1, N$, $\mathbf{1}_{i,j}$ is 1 if the wind direction for grid cell (i, j) is attributed to wind sector n, and $w_{i,j}$ is the corresponding weight for grid cell (i, j) with $i = 1, I$ (longitude) and $j = 1, J$ (latitude).

Thus each wind sector $n = 1, N$ gets a weighted frequency $f_1, \ldots, f_N$. The sector with the highest frequency f_n will be chosen as main wind direction sector, but only if the frequency f_n exceeds a certain threshold (`maindirthreshold`, see Sec. 4). Or in other words: n is chosen as the main wind direction threshold for the sector with the highest $f_\bullet$ if and only if the weighted frequency $f_\bullet$ fulfills:

$$\frac{f_\bullet}{\sum_{i=1}^{I} \sum_{j=1}^{J} w_{i,j}} > \text{maindirthreshold} \tag{6}$$

... where `maindirthreshold` is between 0 and 1. If no main wind sector can be found the WLK classification returns wind sector `0` which indicates "variable wind direction".

3 Input Data

The main method to retrieve the WLK classification is `getClassification(...)` which expects the input data as two NetCDF files. Two files are required as surface level data and pressure level data cannot be mixed (at least not when using `grib_to_netcdf` from the ECMWF GribAPI library I used).

`Rcost` contains two example files which can be loaded via `demofiles()`:

```
> ## Loading demo data (as NetCDF)
> library("Rcost")
> nc <- demofiles()

Surface file:        /tmp/Rtmp7vpbHs/Rinst61cc1f9872ed/Rcost/extdata/ITERIM_sfc.nc
Pressure level file: /tmp/Rtmp7vpbHs/Rinst61cc1f9872ed/Rcost/extdata/ITERIM_pl.nc

> print( names(nc) )

[1] "sfc" "pl"
```

The function `demofiles()` returns a `list` containing the file names (`character` strings) of the two demo files.

4 Parameters and Settings

COST 733 WLK provides a set of "tuning parameters", options, or settings. Not all of them are implemented yet in the `Rcost` package, but—if required—adding them should not be a big deal.

Implemented settings or options:

- `steps`: allows to subset in time (only take some steps out of all time steps provided by the input data files).
- `subset`: an `extent` object to specify an areal subset on which the classification should be performed. If not given, the whole are as provided by the input files will be used.
- `pwclim`: precipitable water climatology (warning: that's the original COST 733 name, while I am currently using total column water content (`tcw`); be aware of this). If given a binary dry/wet class will be returned.
- `weights`: weights for the 'core', 'mid', and 'margin' area (see below).
- `nsector`: number of wind sectors for the classification.
- `maindirthreshold`: threshold of (weighted) grid cells to have the same wind direction for a wind sector to be choosen. If none of the wind sectors exceed this threshold no main wind direction will be classified (variable).
- `zamg`: special mode as used by the ZAMG[1]. Uses pre-specified `subset`, `weights`, and tailored wind sectors for the European Alps.

4.1 Weights and Stripsize

The classification uses a (squared) centered weighting mask for the area of interest. This is done by specifying a weight mask, typically done internally (in `getClassification(...)` given the inputs.

For demonstration purposes these masks can also be created manually using the `weightmask(...)` function of the package. The extent is here defined by `lon` and `lat`, the weights for the three different segments by vector `weights`. `stripsize` can be used to change the weighting scheme. By default `stripsize=5`: the area is split by 5 along longitude and latitude. The outer $1/5th$ of the area then belongs to 'margin', the next $1/5th$ to 'mid' and $1/5 \times 1/5$ defines the core.

Note that, if you set `getClassification(...,zamg=TRUE)` `weightmaskZAMG(...)` instead of `weightmask(...)` is used.

Warning: please note that the `zamg = TRUE` mode works with a hardcoded sub-region! This mode is designed for the Alps and can only be applied there. Let's try what happens if we try to use a sub-region different than 1.5E40.5N to 25.5E54.5N with `weightmaskZAMG(...)`:

[1] Zentralanstalt für Meteorologie und Geodynamik, Vienna, Austria

```
> ## Helper function: Converts weight mask into raster
> wm2raster <- function( x, grid, name ) {
+    attach(grid); delta <- median(diff(lon))
+    res <- raster( x[nrow(x):1,], xmn=min(lon)-delta/2, xmx=max(lon)+delta/2,
+                                  ymn=min(lat)-delta/2, ymx=max(lat)+delta/2 )
+    names(res) <- name; return( res )
+ }
> ## Sepcify weight mask for a grid 10E40N to 20E50N with
> ## a grid spacing of 0.25 degrees (regular ll)
> grid <- list( lon = seq(20, 40, by = 0.25),
+               lat = seq(30, 50, by = 0.25) )
> ## Specify the weights for:
> ## - core segment
> ## - mid segment
> ## - margin segment
> weights <- c(15,2,1)
> ## Get weight mask
> w1 <- weightmask( grid$lon, grid$lat, weights )
> ## We can, of course, also use different weights
> w2 <- weightmask( grid$lon, grid$lat, c(10,5,3) )
> ## Or different stripsize
> w3 <- weightmask( grid$lon, grid$lat, c(10,5,3), stripsize = 7 )
> w4 <- weightmask( grid$lon, grid$lat, c(10,5,3), stripsize = 11 )
> ## Create a RasterStack object for plotting purposes.
> library("raster")
> res <- stack( wm2raster( w1, grid, "weightmask_w1" ),
+               wm2raster( w2, grid, "weightmask_w1" ),
+               wm2raster( w3, grid, "weightmask_w1" ),
+               wm2raster( w4, grid, "weightmask_w1" ) )
> ## Plot
> plot( res, nc = 4, col=rev(terrain.colors(17))[3:17], asp = NA )
```

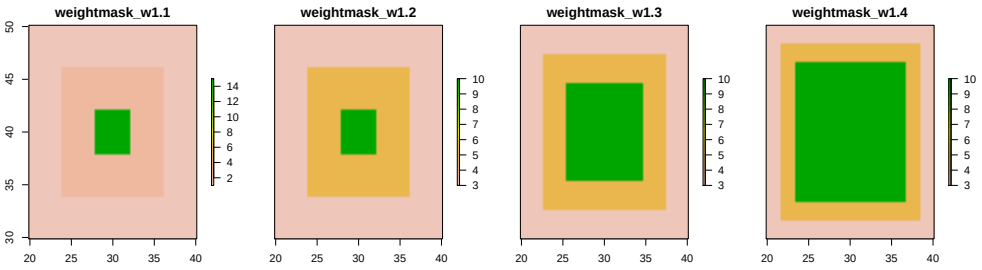

Figure 1: Example of four different weighting mask specifications. Typically done inside the getClassification method.

5 Run Classification

The only function needed (beside `nc_open` and `nc_close` from the `ncdf4` package) is `getClassification`. `getClassification` takes care of identifying the correct fields in the input files, the spatial extent and the subsetting (if required), and the weights.

```
> ## Getting reuqired packages
> library("Rcost"); library("ncdf4")
```

```
> ## Create RasterStack again, once with the default weight mask,
> ## once with the weight mask for the ZAMG mode.
> ## WARNING: zamg = TRUE uses 1.5E40.5N to 25.5E54.5N (hardcoded)
> grid <- list( lon = seq( 1.5, 25.5, by = 0.5),
+               lat = seq(40.5, 54.5, by = 0.5) )
> w1 <- weightmask( grid$lon, grid$lat )
> wZ <- weightmaskZAMG( grid$lon, grid$lat )
> ## Create a RasterStack object for plotting purposes.
> res <- stack( wm2raster( w1, grid, "weight_cost" ),
+               wm2raster( wZ, grid, "weight_ZAMG" ) )
> names(res) <- c("weightmask","weightmask_ZAMG")
> ## Plot
> plot( res, nc = 2, col=rev(terrain.colors(17))[3:17], asp = NA )
```

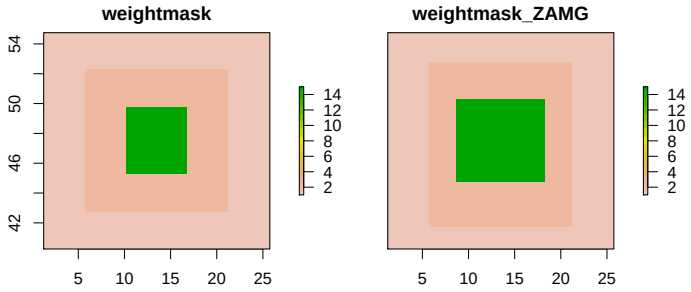

Figure 2: Weight mask using the cost weightmask function (left) and ZAMG weightmask function (right). Note that the ZAMG weight mask uses hardcoded longitude/latitude limits.

```
> ## Create RasterStack again, once with the default weight mask,
> ## once with the weight mask for the ZAMG mode.
> grid <- list( lon = seq( 10, 30, by = 0.5),
+               lat = seq( 45, 60, by = 0.5) )
> w1 <- weightmask( grid$lon, grid$lat )
> wZ <- weightmaskZAMG( grid$lon, grid$lat )
> ## Create a RasterStack object for plotting purposes.
> res <- stack( wm2raster( w1, grid, "weightmask_cost" ),
+               wm2raster( wZ, grid, "weightmask_ZAMG" ) )
> ## Plot
> plot( res, nc = 2, col=rev(terrain.colors(17))[3:17], asp = NA )
```

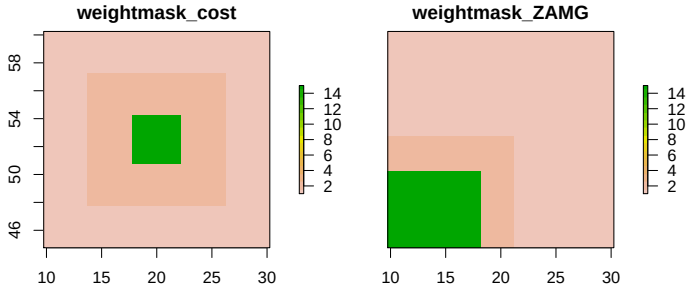

Figure 3: Weight mask using the cost weightmask function (left) and ZAMG weightmask function (right). Note that the ZAMG weight mask uses hardcoded longitude/latitude limits and is not centered anymore!

```
> ## Get demo file location
> files <- demofiles()

Surface file:        /tmp/Rtmp7vpbHs/Rinst61cc1f9872ed/Rcost/extdata/ITERIM_sfc.nc
Pressure level file: /tmp/Rtmp7vpbHs/Rinst61cc1f9872ed/Rcost/extdata/ITERIM_pl.nc

> ## Open NetCDF file connection
> nc <- list( nc_open(files$sfc), nc_open(files$pl) )
> ## Extract correct initialization date/time (used to properly
> ## specify the output object)
> init <- as.POSIXct( ncvar_get(nc[[1]],"time")[1]*3600, origin="1900-01-01" )
> x  <- getClassification( nc, init )
> ## Close NetCDF files
> for ( n in nc ) nc_close( n )
```

6 Output

The output variable `x` contains the COST 733 classification. Namely the columns:

```
> print( x )

                    yday sector cyclonic925 cyclonic500 wet        c925
2000-07-01 12:00:00  182      0           0           0 -99 -0.64589494
2000-07-02 12:00:00  183      0           0           0 -99 -1.10408242
2000-07-03 12:00:00  184      0           0           0 -99 -0.74584787
2000-07-04 12:00:00  185      0           0           0 -99 -0.40608342
```

```
2000-07-05 12:00:00  186      6           0           0 -99 -0.08755121
                          c500      pwi
2000-07-01 12:00:00 -0.1485894 21.11848
2000-07-02 12:00:00 -0.2710856 21.10236
2000-07-03 12:00:00 -0.3532046 21.69227
2000-07-04 12:00:00 -0.3416151 21.13643
2000-07-05 12:00:00 -0.3650751 19.55777
attr(,"init")
[1] "2000-07-01"
attr(,"created")
[1] "2017-10-04 05:21:49 UTC"
```

The returned object `x` is of type `zoo` (see `zoo` package manual) containing the following information:

- `yday`: day of the year, $[0 - 364]$ (analog to the R `POSIXlt` specification) where `yday = 0` is the first of January.
- `sector`: main wind sector, or `0` if no wind sector was found.
- `cyclonic925`/`cyclonic500`: binary class whether the current circulation is anticyclonic (`cyclonicXXX = 0`) or cyclonic (`cyclonicXXX = 1`) on the 925 hPa and 500 hPa surface, respectively.
- `cyclonic500`: same as `cyclonic925` but for the 500 hPa level.
- `wet`: binary response whether this time step is dry (`wet = 0`) or wet (`wet = 1`) compared to a climatology. As we haven't specified `pwclim` when calling `getClassification(...)` all values will be `-99` (classification not possible).
- `c925`/`c500`: numeric value on which `cyclonic925` and `cyclonic500` are based.
- `pwi`: numeric value on which `wet` is based (if `pwclim` is given). Note: still based on total column water (`tcw`).

As we have not specified a subset the classification `x` is based on the whole domain as provided by the demo files (-15.0E15.0N to 35.2E65.2N).

Appendix/Examples

The `Rcost getClassification` function provides a variety of input arguments to adjust the classification and customize it. Some examples:

```
> files <- demofiles(TRUE)
> nc <- list(nc_open(files$sfc),nc_open(files$pl))
> init <- as.POSIXct( ncvar_get(nc[[1]],"time")[1]*3600, origin="1900-01-01" )
```

Step-Subset

```
> ## Only the first two steps: +12h and +36h
> ## Not sure why I count from 00 UTC from the init??
> getClassification( nc, init, steps = c(12,36) )
```

```
                    yday sector cyclonic925 cyclonic500 wet       c925
2000-07-01 12:00:00  182      0           0           0 -99 -0.6458949
2000-07-02 12:00:00  183      0           0           0 -99 -1.1040824
                          c500      pwi
2000-07-01 12:00:00 -0.1485894 21.11848
2000-07-02 12:00:00 -0.2710856 21.10236
attr(,"init")
[1] "2000-07-01"
attr(,"created")
[1] "2017-10-04 05:21:49 UTC"
```

Provide pw climatology (creates binary dry/wet)

```
> ## Specify a subset (ZAMG subset) and use ZAMG mode
> pwclim <- rep(24.5,365) # demo: constant clim for the whole year
> getClassification( nc, init, pwclim = pwclim )
```

```
                    yday sector cyclonic925 cyclonic500 wet        c925
2000-07-01 12:00:00  182      0           0           0   0 -0.64589494
2000-07-02 12:00:00  183      0           0           0   0 -1.10408242
2000-07-03 12:00:00  184      0           0           0   0 -0.74584787
2000-07-04 12:00:00  185      0           0           0   0 -0.40608342
2000-07-05 12:00:00  186      6           0           0   0 -0.08755121
                          c500      pwi
2000-07-01 12:00:00 -0.1485894 21.11848
2000-07-02 12:00:00 -0.2710856 21.10236
2000-07-03 12:00:00 -0.3532046 21.69227
2000-07-04 12:00:00 -0.3416151 21.13643
2000-07-05 12:00:00 -0.3650751 19.55777
attr(,"init")
[1] "2000-07-01"
attr(,"created")
[1] "2017-10-04 05:21:49 UTC"
```

Different number of wind sectors

```
> ## Only four wind sectors
> getClassification( nc, init, nsector = 4 )
```

```
                    yday sector cyclonic925 cyclonic500 wet        c925
2000-07-01 12:00:00  182      0           0           0 -99 -0.64589494
2000-07-02 12:00:00  183      0           0           0 -99 -1.10408242
2000-07-03 12:00:00  184      3           0           0 -99 -0.74584787
2000-07-04 12:00:00  185      3           0           0 -99 -0.40608342
2000-07-05 12:00:00  186      3           0           0 -99 -0.08755121
                          c500      pwi
2000-07-01 12:00:00 -0.1485894 21.11848
2000-07-02 12:00:00 -0.2710856 21.10236
2000-07-03 12:00:00 -0.3532046 21.69227
2000-07-04 12:00:00 -0.3416151 21.13643
2000-07-05 12:00:00 -0.3650751 19.55777
attr(,"init")
[1] "2000-07-01"
attr(,"created")
[1] "2017-10-04 05:21:49 UTC"
```

Spatial subset

```
> ## Specify a subset (using extent)
> getClassification( nc, init, subset = extent(c(1.5,25.5,40.5,54.5)) )

                    yday sector cyclonic925 cyclonic500 wet       c925
2000-07-01 12:00:00  182      6           0           0 -99 -0.9588802
2000-07-02 12:00:00  183      6           1           0 -99  0.3598272
2000-07-03 12:00:00  184      6           1           0 -99  0.8748252
2000-07-04 12:00:00  185      6           1           1 -99  5.5662193
2000-07-05 12:00:00  186      6           0           1 -99 -1.9555651
                          c500      pwi
2000-07-01 12:00:00 -0.3433479 23.56324
2000-07-02 12:00:00 -0.4942587 25.19957
2000-07-03 12:00:00 -0.6704951 27.66346
2000-07-04 12:00:00  0.2737864 26.65847
2000-07-05 12:00:00  0.2644568 21.13933
attr(,"init")
[1] "2000-07-01"
attr(,"created")
[1] "2017-10-04 05:21:50 UTC"
```

Spatial subset in combination with ZAMG mode

```
> ## Specify a subset (ZAMG subset) and use ZAMG mode
> getClassification( nc, init, subset = extent(c(1.5,25.5,40.5,54.5)), zamg = TRUE )

                    yday sector cyclonic925 cyclonic500 wet       c925
2000-07-01 12:00:00  182      7           0           0 -99 -1.3235325
2000-07-02 12:00:00  183      7           0           0 -99 -0.3347344
2000-07-03 12:00:00  184      6           1           0 -99  0.2670699
2000-07-04 12:00:00  185      6           1           1 -99  5.4030935
2000-07-05 12:00:00  186      7           0           1 -99 -3.0763223
                          c500      pwi
2000-07-01 12:00:00 -0.3579545 23.57108
2000-07-02 12:00:00 -0.5592651 25.34477
2000-07-03 12:00:00 -0.7035906 27.97353
2000-07-04 12:00:00  0.3477358 26.61799
2000-07-05 12:00:00  0.2661390 20.95320
attr(,"init")
[1] "2000-07-01"
attr(,"created")
[1] "2017-10-04 05:21:50 UTC"
```

Use custom weights

```
> ## Specify a subset (ZAMG subset) and use ZAMG mode
> getClassification( nc, init, weights = c(10,5,4) )

                    yday sector cyclonic925 cyclonic500 wet      c925
2000-07-01 12:00:00  182      0           1           0 -99 0.6204215
2000-07-02 12:00:00  183      0           1           0 -99 0.2502185
2000-07-03 12:00:00  184      0           1           0 -99 0.4006463
2000-07-04 12:00:00  185      0           1           0 -99 0.1354467
2000-07-05 12:00:00  186      0           1           0 -99 0.1053629
```

```
                          c500      pwi
2000-07-01 12:00:00 -0.03758110 20.55323
2000-07-02 12:00:00 -0.07662597 20.84237
2000-07-03 12:00:00 -0.09613278 20.56827
2000-07-04 12:00:00 -0.12897564 20.19574
2000-07-05 12:00:00 -0.15800884 19.97140
attr(,"init")
[1] "2000-07-01"
attr(,"created")
[1] "2017-10-04 05:21:50 UTC"
```

Higher 'main wind sector' frequency threshold

```
> ## Specify a subset (ZAMG subset) and use ZAMG mode
> getClassification( nc, init, maindirthreshold = 0.8 )

                    yday sector cyclonic925 cyclonic500 wet        c925
2000-07-01 12:00:00  182      0           0           0 -99 -0.64589494
2000-07-02 12:00:00  183      0           0           0 -99 -1.10408242
2000-07-03 12:00:00  184      0           0           0 -99 -0.74584787
2000-07-04 12:00:00  185      0           0           0 -99 -0.40608342
2000-07-05 12:00:00  186      0           0           0 -99 -0.08755121
                         c500      pwi
2000-07-01 12:00:00 -0.1485894 21.11848
2000-07-02 12:00:00 -0.2710856 21.10236
2000-07-03 12:00:00 -0.3532046 21.69227
2000-07-04 12:00:00 -0.3416151 21.13643
2000-07-05 12:00:00 -0.3650751 19.55777
attr(,"init")
[1] "2000-07-01"
attr(,"created")
[1] "2017-10-04 05:21:50 UTC"
```