StormR

Introduction

This vignette teaches you how to use StormR package with illustrations of few basics examples. The main characteristics and features of the package's functionalities are enlighten but we strongly advise the reader to check on the documentation for a deeper insight of the whole package.

Basic workflow

From now on, we will demonstrate what must/may be done step by step throughout the different sub sections, in order to use this package in its best possible way.

Choose a StormsDataset

The first thing to do after loading the package is to initialize a database for Tropical Depressions/Storms/Cyclones (TD/TS/TC). This will let the package know which database the user wants to extract TD/TS/TC from. We higlhy recommand using IBTRaCS databases as they provide the most extensive and relevant informations about TD/TS/TC overall. Thus, this operation is carried out calling the initDatabase function whose prototype with the default setting is showed right here:

filename input must contain the path to the database the user is interested in. Currently, this package can only read netcdf files (.nc). If one is interested in using a database with another format (usually csv), note that several tools exist to convert file in the netcdf format. Also, a dictionary input (field) will inform which are the dimensions in the netcdf file that contain the desired data. Note that some fields are mandatory (names, seasons, lon, lat, isoTime, msw) while others are recommended (rmw, sshs) or highly recommended (pressure, poci, basin). Last but not least, a basin input can be used as a filter so that only TD/TS/TC that occured in the specified basin are extracted from the database. We strongly recommend using this input as it may considerably speed up the next functions.

This function returns in fine a StormsDataset object, especially designed to gathers all these above informations about the database.

In what follows, the StormsDataset used relies on the IBTrACS.SP.v04r00.nc. This dataset gathers all TD/TS/TC that occured in the South Pacific ocean since 1980. This dataset named IBTRACS_SP is actually the default used in the package's functions (See next section).

Get data associated with storms

Once the StormsDataset is loaded, the next step is to select storms we are interested in. This operation is done using getStorms function. It collects the TD/TS/TC coming from a StormsDataset (via sds input) over a certain Location Of Interest (loi input). This area is extended using the max_dist input afterwards. Default value is set to 300km. Note that sds and loi inputs are mandatory to perform this function (See getStorms Documentation). It is also possible to filter TD/TS/TC by their cyclonic seasons and/or names (season and input names). Finally, storms with maximum wind speed lower than 18 m/s (Tropical Depressions in the Saffir Simpson Hurricane Scale SSHS) can be ignored using remove_TD logical input. Default value is set to TRUE.

Here are some basic usage of the getStorms function. In this case, we get the data associated with the tropical cyclone Harold that hit Vanuatu in 2020. Note here that the loi represents a whole country. (See getStorms

documentation to get the full list of country available).

```
harold <- getStorms(loi = "Vanuatu", names = "HAROLD")
```

```
## === getStorms processing ... ===
##
## -> Making buffer: Done
## -> Searching for HAROLD storm ...
##    -> Identifying Storms: Done
## -> Gathering storm(s) ...
##   |                                                                              |
##
## === DONE with run time 2.763335 sec ===
##
## SUMMARY:
## (*) LOI: Vanuatu
## (*) Buffer size: 300 km
## (*) Remove Tropical Depressions (< 18 m/s in sshs): yes
## (*) Number of Storms: 1
##         Name - Tropical season - SSHS - Number of observation within buffer:
##         HAROLD - 2020 - 5 - 26
```

In this second example, we collect data for all tropical storms and cyclones over the Exclusive Economic Zone of New Caledonia (eezNC) between 2000 and 2022. The loi here is a sf object, but it can also be a shapefile.

```
sts.nc <- getStorms(loi = eezNC, seasons = c(2000,2022))
```

```
## === getStorms processing ... ===
##
## -> Making buffer: Done
## -> Searching storms from 2000 to 2022 ...
##    -> Identifying Storms: 185 potential candidates...
## -> Gathering storm(s) ...
##   |                                                                              |
```

```
##         KERRY - 2005 - 2 - 63
##         JIM - 2006 - 1 - 25
##         LARRY - 2006 - 4 - 16
##         WATI - 2006 - 1 - 48
##         YANI - 2007 - 1 - 58
##         BECKY - 2007 - 1 - 31
##         FUNA - 2008 - 3 - 28
##         GENE - 2008 - 3 - 52
##         INNIS - 2009 - 0 - 16
##         HAMISH - 2009 - 4 - 13
##         JASPER - 2009 - 0 - 19
##         RENE - 2010 - 3 - 5
##         ULUI - 2010 - 5 - 59
##         VANIA - 2011 - 0 - 39
##         ZELIA - 2011 - 2 - 13
##         WILMA - 2011 - 4 - 14
##         ANTHONY - 2011 - 0 - 37
##         YASI - 2011 - 4 - 10
##         ATU - 2011 - 4 - 19
##         JASMINE - 2012 - 4 - 49
##         DAPHNE - 2012 - 0 - 11
##         FREDA - 2013 - 3 - 42
##         SANDRA - 2013 - 3 - 49
##         JUNE - 2014 - 0 - 21
##         EDNA - 2014 - 0 - 19
##         HADI - 2014 - 0 - 1
##         LUSI - 2014 - 1 - 29
##         ITA - 2014 - 5 - 11
##         OLA - 2015 - 2 - 36
##         MARCIA - 2015 - 4 - 11
##         PAM - 2015 - 5 - 13
##         SOLO - 2015 - 0 - 29
##         ULA - 2016 - 4 - 46
##         TATIANA - 2016 - 0 - 31
##         WINSTON - 2016 - 5 - 79
##         ZENA - 2016 - 2 - 12
##         COOK - 2017 - 2 - 33
##         DONNA - 2017 - 4 - 44
##         FEHI - 2018 - 0 - 28
##         GITA - 2018 - 4 - 31
##         HOLA - 2018 - 3 - 33
##         LINDA - 2018 - 0 - 30
##         IRIS - 2018 - 0 - 66
##         JOSIE - 2018 - 0 - 7
##         OWEN - 2019 - 2 - 13
##         PENNY - 2019 - 0 - 22
##         OMA - 2019 - 1 - 126
##         ANN - 2019 - 0 - 19
##         UESI - 2020 - 1 - 43
##         GRETEL - 2020 - 1 - 19
##         HAROLD - 2020 - 5 - 25
##         LUCAS - 2021 - 1 - 29
##         NIRAN - 2021 - 5 - 16
##         RUBY - 2022 - 1 - 28
```

```
##         SETH - 2022 - 0 - 25
##         CODY - 2022 - 0 - 31
##         DOVI - 2022 - 2 - 21
##         FILI - 2022 - 0 - 43
##         GINA - 2022 - 0 - 24
```

In this last example, we retrieve all data associated with tropical storms and cyclones that occured since 1980 around the point coordinate 188.17:-13.92 (longitude, latitude decimal degree) within a 300km buffer. These coordinates are actually located in the American Samoa.

```
pt <- c(188.17,-13.92)
sts.pt <- getStorms(loi = pt)
```

```
## === getStorms processing ... ===
##
## -> Making buffer: Done
## -> Searching storms from 1980 to 2022 ...
##    -> Identifying Storms: 386 potential candidates...
## -> Gathering storm(s) ...
##   |                                                                              |
```

```
##         ELLA - 2017 - 1 - 14
##         GITA - 2018 - 4 - 7
##         VICKY - 2020 - 0 - 10
##         WASI - 2020 - 0 - 13
```

Access data

The getStorms function returns data collected from TD/TS/TC in a Storms object especially designed for this purpose (See Storms class). Then , one can be interested in getting basics informations from a Storms object initialized with getStorms. However the structure of this object is quite complex and it can rapidly become overwhelming trying to reach data on your own. Here are some getters that will help you saving time to access data.

From now on, we demonstrate how to use them using the sts.nc Storms object initialized right above. First of all, if you are interested in getting all the storm names, just run the following getter:

```
getNames(sts.nc)
```

```
##  [1] "IRIS"    "JO"      "VAUGHAN" "PAULA"   "SOSE"    "CLAUDIA" "DES"
##  [8] "ZOE"     "BENI"    "ERICA"   "ESETA"   "GINA"    "IVY"     "GRACE"
## [15] "KERRY"   "JIM"     "LARRY"   "WATI"    "YANI"    "BECKY"   "FUNA"
## [22] "GENE"    "INNIS"   "HAMISH"  "JASPER"  "RENE"    "ULUI"    "VANIA"
## [29] "ZELIA"   "WILMA"   "ANTHONY" "YASI"    "ATU"     "JASMINE" "DAPHNE"
## [36] "FREDA"   "SANDRA"  "JUNE"    "EDNA"    "HADI"    "LUSI"    "ITA"
## [43] "OLA"     "MARCIA"  "PAM"     "SOLO"    "ULA"     "TATIANA" "WINSTON"
## [50] "ZENA"    "COOK"    "DONNA"   "FEHI"    "GITA"    "HOLA"    "LINDA"
## [57] "IRIS"    "JOSIE"   "OWEN"    "PENNY"   "OMA"     "ANN"     "UESI"
## [64] "GRETEL"  "HAROLD"  "LUCAS"   "NIRAN"   "RUBY"    "SETH"    "CODY"
## [71] "DOVI"    "FILI"    "GINA"
```

Also, for each storms in your Storms object, the following getters will respectively return the cyclonic season and the maximum category reached in the SSHS:

```
#Get cyclonic seasons
getSeasons(sts.nc)
```

```
##    IRIS      JO VAUGHAN   PAULA    SOSE CLAUDIA     DES     ZOE    BENI   ERICA
##    2000    2000    2000    2001    2001    2002    2002    2003    2003    2003
##   ESETA    GINA     IVY   GRACE   KERRY     JIM   LARRY    WATI    YANI   BECKY
##    2003    2003    2004    2004    2005    2006    2006    2006    2007    2007
##    FUNA    GENE   INNIS  HAMISH  JASPER    RENE    ULUI   VANIA   ZELIA   WILMA
##    2008    2008    2009    2009    2009    2010    2010    2011    2011    2011
## ANTHONY    YASI     ATU JASMINE  DAPHNE   FREDA  SANDRA    JUNE    EDNA    HADI
##    2011    2011    2011    2012    2012    2013    2013    2014    2014    2014
##    LUSI     ITA     OLA  MARCIA     PAM    SOLO     ULA TATIANA WINSTON    ZENA
##    2014    2014    2015    2015    2015    2015    2016    2016    2016    2016
##    COOK   DONNA    FEHI    GITA    HOLA   LINDA    IRIS   JOSIE    OWEN   PENNY
##    2017    2017    2018    2018    2018    2018    2018    2018    2019    2019
##     OMA     ANN    UESI  GRETEL  HAROLD   LUCAS   NIRAN    RUBY    SETH    CODY
##    2019    2019    2020    2020    2020    2021    2021    2022    2022    2022
##    DOVI    FILI    GINA
##    2022    2022    2022
```

```
#Get maximum reached category in SSHS
getSSHS(sts.nc)
```

```
##    IRIS      JO VAUGHAN   PAULA    SOSE CLAUDIA     DES     ZOE    BENI   ERICA
```

```
##       1       1       0       3       1       1       0       5       4       4
##   ESETA    GINA     IVY   GRACE   KERRY     JIM   LARRY    WATI    YANI   BECKY
##       3       2       3       0       2       1       4       1       1       1
##    FUNA    GENE   INNIS  HAMISH  JASPER    RENE    ULUI   VANIA   ZELIA   WILMA
##       3       3       0       4       0       3       5       0       2       4
## ANTHONY    YASI     ATU JASMINE  DAPHNE   FREDA  SANDRA    JUNE    EDNA    HADI
##       0       4       4       4       0       3       3       0       0       0
##    LUSI     ITA     OLA  MARCIA     PAM    SOLO     ULA TATIANA WINSTON    ZENA
##       1       5       2       4       5       0       4       0       5       2
##    COOK   DONNA    FEHI    GITA    HOLA   LINDA    IRIS   JOSIE    OWEN   PENNY
##       2       4       0       4       3       0       0       0       2       0
##     OMA     ANN    UESI  GRETEL  HAROLD   LUCAS   NIRAN    RUBY    SETH    CODY
##       1       0       1       1       5       1       5       1       0       0
##    DOVI    FILI    GINA
##       2       0       0
```

This getter simply returns the number of storms provided in your Storms Object:

```
getNbStorms(sts.nc)
```

```
## [1] 73
```

In addition, the next 3 getters are useful to retrieve spatial informations on the LOI of your Storms object. The first command will return the LOI converted in sf format:

```
getLOI(sts.nc)
```

```
## Simple feature collection with 1 feature and 0 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 156.2557 ymin: -26.20108 xmax: 174.2757 ymax: -14.82636
## Geodetic CRS:  WGS 84
##                            loi.sf
## 1 POLYGON ((164.1694 -15.9122...
```

This second command simply returns the size (in km) of the buffer used to extent the LOI:

```
getBufferSize(sts.nc)
```

```
## [1] 300
```

Finally this third command provides the LOI extended with the buffer:

```
getBuffer(sts.nc)
```

```
## Simple feature collection with 1 feature and 0 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 153.3918 ymin: -28.97837 xmax: 177.3811 ymax: -12.05625
## Geodetic CRS:  WGS 84
##                            loi.sf
## 1 POLYGON ((168.2932 -16.0239...
```

One can also be interested in getting the overall informations about a particular storm. This operation is achieved using the following getter:

```
niran <- getStorm(sts = sts.nc, name = "NIRAN")
```

Note: If serveral storms share the same name, the cyclonic season must be specified to differentiate them. For example, 2 storms named Evan are provided within the sts.pt Storms object initialized in the first section.

This first command will then throw an error, as we did not specify which one we are interested in.

```
#getStorm(s = sts.pt, name = "EVAN")
```

We thus tackle this issue using the next 2 commands:

```
evan1997 <- getStorm(s = sts.pt, name = "EVAN", season = 1997)
evan2013 <- getStorm(s = sts.pt, name = "EVAN", season = 2013)
```

All these getters are designed to retrieve general informations on first levels of Storms objects. However we can go further into the object getting data of a particular storm. In that way, the following getters work with both Storms and Storm signature (See Storm class).

These commands are simply getNames, getSeasons and getSSHS calls for signature Storm object:

```
getNames(niran)
```

```
## [1] "NIRAN"
```

```
#Equivalent to getNames(s = sts.nc, name = "NIRAN")
getSeasons(niran)
```

```
## [1] 2021
```

```
#Equivalent to getSeasons(s = sts.nc, name = "NIRAN")
getSSHS(niran)
```

```
## [1] 5
```

```
#Equivalent to getSSHS(s = sts.nc, name = "NIRAN")
```

To conclude, these 3 last getters provide information about observations for a particular storm. They respectively return the number of observations, all of the observations and the indices of observations within the extended LOI for a particular storm, here tropical cyclone Niran (2021) over New Caledonia:

```
getNbObs(niran)
```

```
## [1] 53
```

```
#Equivalent to getNbObs(s= sts.nc, name = "NIRAN")
getObs(niran)
```

```
##               iso.time      lon       lat msw sshs rmw pres poci
## 1  2021-03-01 00:00:00 147.2000 -17.20000  18    0  74  996 1004
## 2  2021-03-01 03:00:00 146.9599 -16.95511  18    0  74  995 1002
## 3  2021-03-01 06:00:00 146.8000 -16.70000  18    0  74  994 1000
## 4  2021-03-01 09:00:00 146.7825 -16.42242  19    0  69  994 1001
## 5  2021-03-01 12:00:00 146.9000 -16.10000  20    0  65  994 1002
## 6  2021-03-01 15:00:00 147.1277 -15.69257  22    0  50  989 1002
## 7  2021-03-01 18:00:00 147.4000 -15.30000  23    0  37  984 1002
## 8  2021-03-01 21:00:00 147.6225 -14.99000  23    0  50  988 1002
## 9  2021-03-02 00:00:00 147.8000 -14.80000  23    0  65  993 1003
## 10 2021-03-02 03:00:00 147.9149 -14.76247  25    0  56  991 1003
## 11 2021-03-02 06:00:00 148.0000 -14.80000  26    0  46  990 1003
## 12 2021-03-02 09:00:00 148.0850 -14.79247  27    0  50  988 1003
## 13 2021-03-02 12:00:00 148.2000 -14.80000  28    0  56  986 1004
## 14 2021-03-02 15:00:00 148.4412 -14.85238  28    0  56  985 1002
## 15 2021-03-02 18:00:00 148.6000 -14.90000  28    0  56  984 1001
## 16 2021-03-02 21:00:00 148.4812 -14.89725  30    0  56  980 1002
## 17 2021-03-03 00:00:00 148.2499 -14.86911  32    0  56  977 1004
```

```
## 18 2021-03-03 03:00:00 148.0187 -14.83226  34    1  46  973 1004
## 19 2021-03-03 06:00:00 147.9000 -14.80000  36    1  37  970 1004
## 20 2021-03-03 09:00:00 148.0738 -14.76490  36    1  37  970 1004
## 21 2021-03-03 12:00:00 148.3000 -14.80000  36    1  37  970 1005
## 22 2021-03-03 15:00:00 148.3001 -14.99995  37    1  30  970 1003
## 23 2021-03-03 18:00:00 148.3000 -15.20000  38    1  22  970 1001
## 24 2021-03-03 21:00:00 148.4401 -15.20742  41    1  24  966 1001
## 25 2021-03-04 00:00:00 148.7000 -15.20000  43    2  26  962 1001
## 26 2021-03-04 03:00:00 149.0622 -15.35752  44    2  24  960 1000
## 27 2021-03-04 06:00:00 149.5000 -15.50000  46    2  22  958 1000
## 28 2021-03-04 09:00:00 149.8532 -15.43415  44    2  22  961 1001
## 29 2021-03-04 12:00:00 150.4000 -15.40000  43    2  22  964 1002
## 30 2021-03-04 15:00:00 151.3895 -15.60594  45    2  22  961 1002
## 31 2021-03-04 18:00:00 152.5000 -15.90000  48    2  22  958 1002
## 32 2021-03-04 21:00:00 153.3144 -16.08086  52    3  22  951 1002
## 33 2021-03-05 00:00:00 154.2000 -16.40000  56    3  22  944 1002
## 34 2021-03-05 03:00:00 155.4535 -17.06325  58    3  15  942 1000
## 35 2021-03-05 06:00:00 156.8000 -17.80000  59    4   9  940  999
## 36 2021-03-05 09:00:00 157.9741 -18.32137  65    4   9  928 1000
## 37 2021-03-05 12:00:00 159.1000 -18.80000  71    5   9  917 1001
## 38 2021-03-05 15:00:00 160.2510 -19.32679  68    4   9  921 1000
## 39 2021-03-05 18:00:00 161.4000 -19.90000  66    4   9  925  999
## 40 2021-03-05 21:00:00 162.5068 -20.51831  58    4   9  939  999
## 41 2021-03-06 00:00:00 163.7000 -21.20000  51    3   9  953  999
## 42 2021-03-06 03:00:00 165.1059 -21.92737  51    3   9  952  999
## 43 2021-03-06 06:00:00 166.6000 -22.70000  51    3   9  952 1000
## 44 2021-03-06 09:00:00 167.9737 -23.42875  48    2   9  955 1000
## 45 2021-03-06 12:00:00 169.5000 -24.30000  46    2   9  959 1000
## 46 2021-03-06 15:00:00 171.5424 -25.57994  39    1   9  968 1001
## 47 2021-03-06 18:00:00 173.5000 -26.70000  33    1   9  977 1002
## 48 2021-03-06 21:00:00 174.6781 -27.05317  29    0  59  984 1002
## 49 2021-03-07 00:00:00 175.7000 -27.30000  26    0 111  991 1003
## 50 2021-03-07 03:00:00 177.1815 -28.01794  26    0 111  988 1003
## 51 2021-03-07 06:00:00 178.8000 -28.90000  26    0 111  985 1003
## 52 2021-03-07 09:00:00 180.3689 -29.76719  24    0 111  986 1003
## 53 2021-03-07 12:00:00 181.9000 -30.60000  23    0 111  987 1003
```

```
#Equivalent to getObs(s = sts.nc, name = "NIRAN")
getInObs(niran)
```

```
##  [1] 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
```

```
#Equivalent to getInObs(s = sts.nc, name = "NIRAN")
```

Plot data associated with storms

An interesting feature of this package is the plotStorms function which let the user plot track(s) of storm(s) provided in a Storms object over and beyond the LOI, using different settings (See plotStorms documentation to see all the available input). Here are some basic usages of this function.

In this example, we plot tropical cyclone Harold track over the Vanuatu alongside with the labeled observations. Default settings are used to plot labels which are every 24h and on the right side of observations.

```
plotStorms(harold, labels = TRUE)
```

In this second example, we plot tropical cyclone Erica (2003) and Cook (2017), over the EEZ of New Caledonia

alongside with the labeled observations (In this case every ??H).

```
plotStorms(sts.nc, names = c("ERICA", "COOK"), labels = TRUE, by = 12)
```

In this last example, we plot every tropical cyclone that reached category 5 (SSHS) around the American Samoa, alongside with the labeled observations.

```
plotStorms(sts.pt, category = 5, labels = TRUE)
```

Computing rasterized products

The most important feature provided by this package may be by far the stormBehaviour_sp function. Given a Storms object, it allows the user to compute rasterized product(s) for each storm over the LOI. The available products are the Maximum Sustained Wind (MSW) which is the default, the Power Index Dissipation (PDI), the hour exposition for wind greater than a wind speed threshold (Exposure) and finally 2D wind speed/direction structures for each observations (Profiles). The output raster(s) are stacked in a SpatRaster object in WGS84 projection.

Depending on several inputs, the computations are not undertaken the same. The user must specify a method used to regenerate the wind speed structures in order to compute the desired product, but also the asymmetry used and other parameters. To do so, the package provides two methods which are the wind models derived from Holland (1980) REF? and Willoughby et al. (2006) REF? which is the default. The first one relies on both basic cyclonic Physics and parameters fitting according to cyclonic observations while the second one is based on fits performed on cyclonic observations. As TD/TS/TC are usually not symmetrical, it is possible to modify the structure of wind speed according to the formula derived in Boose et al. (2001) REF ?

Outputs product may also differ modifying the following inputs. Spatial resolution for grid rasters can be chosen among 4 options: 30sec, 2.5min which is the default, 5min and 10min degree. This choice has been made to match Wordlclim data so that it should be easily compared ... ??? . Moreover, the time resolution (in hour) used to interpolate observations in the forthcoming computations can also be chosen among 4 options: 1 (60min) which is the default, 0.75 (45min) , 0.5 (30min) and 0.25 (15min). Note that the finer the spatial/time resolution, the slower the routine.

```
prod.harold <- stormBehaviour_sp(harold, product = c("MSW", "PDI", "Exposure"))
```

```
## === stormBehaviour_sp processing ... ===
##
## Computation settings:
##   (*) Time interpolation: Every 60 min
##   (*) Space resolution: 2.5min
##   (*) Method used: Willoughby
##   (*) Product(s) to compute: MSW PDI Exposure
##   (*) Asymmetry used: Boose01
##
## Storm(s):
##   ( 1 )  HAROLD
##
## HAROLD  ( 1 / 1 )
##   |                                                                      |
```

```
##   3     HAROLD_Exposure_18
##   4     HAROLD_Exposure_33
##   5     HAROLD_Exposure_42
##   6     HAROLD_Exposure_49
##   7     HAROLD_Exposure_58
##   8     HAROLD_Exposure_70
```

```
prof.harold <- stormBehaviour_sp(harold, product = "Profiles")
```

```
## === stormBehaviour_sp processing ... ===
##
## Computation settings:
##   (*) Time interpolation: Every 60 min
##   (*) Space resolution: 2.5min
##   (*) Method used: Willoughby
##   (*) Product(s) to compute: Profiles
##   (*) Asymmetry used: Boose01
##
## Storm(s):
##   ( 1 )  HAROLD
##
## HAROLD  ( 1 / 1 )
##   |                                                                                |
```

```
##   27     HAROLD_Profiles_32.2
##   28     HAROLD_Profiles_33
##   29     HAROLD_Profiles_33.1
##   30     HAROLD_Profiles_33.2
##   31     HAROLD_Profiles_34
##   32     HAROLD_Profiles_34.1
##   33     HAROLD_Profiles_34.2
##   34     HAROLD_Profiles_35
##   35     HAROLD_Profiles_35.1
##   36     HAROLD_Profiles_35.2
##   37     HAROLD_Profiles_36
##   38     HAROLD_Profiles_36.1
##   39     HAROLD_Profiles_36.2
##   40     HAROLD_Profiles_37
##   41     HAROLD_Profiles_37.1
##   42     HAROLD_Profiles_37.2
##   43     HAROLD_Profiles_38
##   44     HAROLD_Profiles_38.1
##   45     HAROLD_Profiles_38.2
##   46     HAROLD_Profiles_39
##   47     HAROLD_Profiles_39.1
##   48     HAROLD_Profiles_39.2
##   49     HAROLD_Profiles_40
##   50     HAROLD_Profiles_40.1
##   51     HAROLD_Profiles_40.2
##   52     HAROLD_Profiles_41
##   53     HAROLD_Profiles_41.1
##   54     HAROLD_Profiles_41.2
##   55     HAROLD_Profiles_42
##   56     HAROLD_Profiles_42.1
##   57     HAROLD_Profiles_42.2
##   58     HAROLD_Profiles_43
##   59     HAROLD_Profiles_43.1
##   60     HAROLD_Profiles_43.2
##   61     HAROLD_Profiles_44
##   62     HAROLD_Profiles_44.1
##   63     HAROLD_Profiles_44.2
##   64     HAROLD_Profiles_45
##   65     HAROLD_Profiles_45.1
##   66     HAROLD_Profiles_45.2
##   67     HAROLD_Profiles_46
##   68     HAROLD_Profiles_46.1
##   69     HAROLD_Profiles_46.2
##   70     HAROLD_Profiles_47
##   71     HAROLD_Profiles_47.1
##   72     HAROLD_Profiles_47.2
##   73     HAROLD_Profiles_48
##   74     HAROLD_Profiles_48.1
##   75     HAROLD_Profiles_48.2
##   76     HAROLD_WindDirection_24
##   77     HAROLD_WindDirection_24.1
##   78     HAROLD_WindDirection_24.2
##   79     HAROLD_WindDirection_25
##   80     HAROLD_WindDirection_25.1
```

```
##   81     HAROLD_WindDirection_25.2
##   82     HAROLD_WindDirection_26
##   83     HAROLD_WindDirection_26.1
##   84     HAROLD_WindDirection_26.2
##   85     HAROLD_WindDirection_27
##   86     HAROLD_WindDirection_27.1
##   87     HAROLD_WindDirection_27.2
##   88     HAROLD_WindDirection_28
##   89     HAROLD_WindDirection_28.1
##   90     HAROLD_WindDirection_28.2
##   91     HAROLD_WindDirection_29
##   92     HAROLD_WindDirection_29.1
##   93     HAROLD_WindDirection_29.2
##   94     HAROLD_WindDirection_30
##   95     HAROLD_WindDirection_30.1
##   96     HAROLD_WindDirection_30.2
##   97     HAROLD_WindDirection_31
##   98     HAROLD_WindDirection_31.1
##   99     HAROLD_WindDirection_31.2
##   100     HAROLD_WindDirection_32
##   101     HAROLD_WindDirection_32.1
##   102     HAROLD_WindDirection_32.2
##   103     HAROLD_WindDirection_33
##   104     HAROLD_WindDirection_33.1
##   105     HAROLD_WindDirection_33.2
##   106     HAROLD_WindDirection_34
##   107     HAROLD_WindDirection_34.1
##   108     HAROLD_WindDirection_34.2
##   109     HAROLD_WindDirection_35
##   110     HAROLD_WindDirection_35.1
##   111     HAROLD_WindDirection_35.2
##   112     HAROLD_WindDirection_36
##   113     HAROLD_WindDirection_36.1
##   114     HAROLD_WindDirection_36.2
##   115     HAROLD_WindDirection_37
##   116     HAROLD_WindDirection_37.1
##   117     HAROLD_WindDirection_37.2
##   118     HAROLD_WindDirection_38
##   119     HAROLD_WindDirection_38.1
##   120     HAROLD_WindDirection_38.2
##   121     HAROLD_WindDirection_39
##   122     HAROLD_WindDirection_39.1
##   123     HAROLD_WindDirection_39.2
##   124     HAROLD_WindDirection_40
##   125     HAROLD_WindDirection_40.1
##   126     HAROLD_WindDirection_40.2
##   127     HAROLD_WindDirection_41
##   128     HAROLD_WindDirection_41.1
##   129     HAROLD_WindDirection_41.2
##   130     HAROLD_WindDirection_42
##   131     HAROLD_WindDirection_42.1
##   132     HAROLD_WindDirection_42.2
##   133     HAROLD_WindDirection_43
##   134     HAROLD_WindDirection_43.1
```

```
##   135     HAROLD_WindDirection_43.2
##   136     HAROLD_WindDirection_44
##   137     HAROLD_WindDirection_44.1
##   138     HAROLD_WindDirection_44.2
##   139     HAROLD_WindDirection_45
##   140     HAROLD_WindDirection_45.1
##   141     HAROLD_WindDirection_45.2
##   142     HAROLD_WindDirection_46
##   143     HAROLD_WindDirection_46.1
##   144     HAROLD_WindDirection_46.2
##   145     HAROLD_WindDirection_47
##   146     HAROLD_WindDirection_47.1
##   147     HAROLD_WindDirection_47.2
##   148     HAROLD_WindDirection_48
##   149     HAROLD_WindDirection_48.1
##   150     HAROLD_WindDirection_48.2
```

Computing point wise products

If one is interested in getting the above products computed on a set of point coordinates rather than computing them on raster grids which takes obsiously more time, then the stormBehaviour_pt function should be used. It is actually the stormBehaviour_sp counterpart where products are computed on specific locations and not rasterized. Here are the main differences between these two functions: * The user must specify the set of point coordinate to use in a data.frame where longitude and latitude must be respectively stored in a "lon" and "lat" column. * MSW product is replaced with the Time Series (TS) product which is the default. It generates wind speed time series at the given location and over the whole storm lifecycle. * Profiles product is no longer available as it does not make sense pointwise.

```
luganville.pt <- data.frame(lon = 167.1667 , lat = -15.5333)

ts.luganville <- stormBehaviour_pt(harold, points = luganville.pt)
```

Visualize products

```
plotBehaviour(harold, prod.harold[["HAROLD_MSW"]])
```

```
plotBehaviour(harold, prod.harold[["HAROLD_PDI"]])
```

```
plotBehaviour(harold, prod.harold[["HAROLD_Exposure_58"]])
```

Save product

```
writeRast(prod.harold, path = paste0(tempdir(),"/"))
```