January 11, 2018

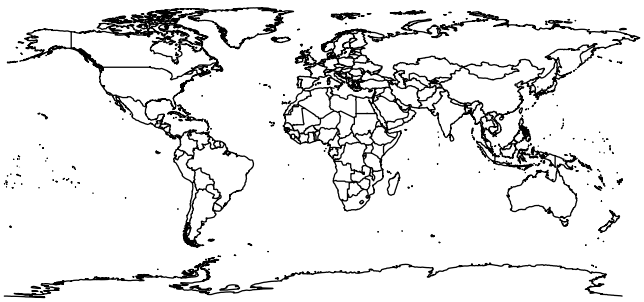
The results below are generated from an R script.

```
library(ncdf4) # For reading the netCDF file
library(RColorBrewer) # For color palttes
library(fields) # For plotting
library(maps) # For basemaps
library(animation) # For making gif
library(dplyr) # For using storms dataset
library(ggmap)  # For maps like library maps



## These will be used for appending the minimum and maximum values for each time step in the loop
xmax<-c() ; xmin <-c()
omax<-c() ; omin<-c()


for (i in seq(90,124,1)) {


fileobj<-nc_open("850_56_42_124.nc") # August 2005, Era Interim Daily Data Set

time<-ncvar_get(fileobj,"time")
Latitude<-ncvar_get(fileobj,"latitude")
Longitude<-ncvar_get(fileobj,"longitude")
u<-ncvar_get(fileobj,"u")
v<-ncvar_get(fileobj,"v")
time_units <- ncatt_get(fileobj,"time","units")


# Flipping Latitudes

u_wind<-array(NA,dim(u))
u_wind[,,]<-u[,ncol(u):1,]
u_wind[,,]<-u[,ncol(u):1,]

v_wind<-array(NA,dim(v))
v_wind[,,]<-v[,ncol(v):1,]
v_wind[,,]<-v[,ncol(v):1,]


uvel=u_wind[,,i]
vvel=v_wind[,,i]
```

```
Latitude1 = Latitude ; Longitude1=Longitude
dtime = c(rep(0,89),"2005-08-23 06:00:00","2005-08-23 12:00:00","2005-08-23 18:00:00","2005-08-24 00:00
          "2005-08-24 06:00:00","2005-08-24 12:00:00","2005-08-24 18:00:00","2005-08-25 00:00:00",
          "2005-08-25 06:00:00","2005-08-25 12:00:00","2005-08-25 18:00:00","2005-08-26 00:00:00",
          "2005-08-26 06:00:00","2005-08-26 12:00:00","2005-08-26 18:00:00","2005-08-27 00:00:00",
          "2005-08-27 06:00:00","2005-08-27 12:00:00","2005-08-27 18:00:00","2005-08-28 00:00:00",
          "2005-08-28 06:00:00","2005-08-28 12:00:00","2005-08-28 18:00:00","2005-08-29 00:00:00",
          "2005-08-29 06:00:00","2005-08-29 12:00:00","2005-08-29 18:00:00","2005-08-30 00:00:00",
          "2005-08-30 06:00:00","2005-08-30 12:00:00","2005-08-30 18:00:00","2005-08-31 00:00:00",
          "2005-08-31 06:00:00","2005-08-31 12:00:00","2005-08-31 18:00:00")


### Vorticity

dx=dy=80000 # resolution of data is about 80km = 80000m

#forward difference for computing dudy and dvdx

dudy = (uvel[,2:42] - uvel[,1:41])/dy

dvdx = (vvel[2:56,] - vvel[1:55,])/dx


# The dimensions of dudy and dvdx are off
# So I add the last row/column to the end+1 so that dimensions will match
# Boundaries are ignored

y = (array(1, dim = c(56,1)))
yy = y*dudy[,41]

x = (array(1, dim = c(1,42)))
xx = x*(t(dvdx[55,]))

dudy = cbind(dudy,yy)

dvdx = rbind(dvdx,xx)


# Vorticity is equal to dvdx-dudy
vort = dvdx-dudy

# To see vorticity max and min
xmax<-append(xmax,max(vort)) ; xmin<-append(xmin,min(vort))


par(mar=c(0.1,0.1,0.1,0.1))

map(database="world", resolution = 0)

jpeg(file=paste("Vorticity_", i, ".jpeg", sep = ""),width=640,height=480)

mycol = c(rev(brewer.pal(9,"Reds")), brewer.pal(8,"Blues"))

mybreaks= c(seq(-0.0001,0.0006,length.out = 18))
```

```
Longitude =Longitude-360 ;  Latitude = rev(Latitude)

image.plot(Longitude,Latitude,vort, breaks = mybreaks , col = mycol,horizontal = T,
           legend.lab = "Vorticity (1/s)")

title(main = paste("Vorticity  " ,dtime[i]) , xlab = "Longitude", ylab = "Latitude")

map(database = "world", resolution = 0, add = T)

dev.off()



#### Okubo Weiss Parameter (1/s^2)   w = dvdx-dudy = vort,  Sn = dudx - dvdy
#### Ss = dvdx + dudy,  Okubo = Sn^2 + Ss^2 - w^2

#forward difference for computing dvdy and dudx
dvdy = (vvel[,2:42] - vvel[,1:41])/dy

dudx = (uvel[2:56,] - uvel[1:55,])/dx


y2 = (array(1, dim = c(56,1)))
yy2 = y2*dvdy[,41]

x2 = (array(1, dim = c(1,42)))
xx2 = x2*(t(dudx[55,]))

dvdy = cbind(dvdy,yy2)

dudx = rbind(dudx,xx2)

Sn = dudx - dvdy
Ss = dvdx + dudy

Okubo = (Sn**2) + (Ss**2) - (vort**2)

# To see maxs ans mins of Okubo Weiss Parameter
omax<-append(omax,max(Okubo)) ; omin<-append(omin,min(Okubo))



par(mar=c(0.1,0.1,0.1,0.1))

map(database="world", resolution = 0)

jpeg(file=paste("Okubo_", i, ".jpeg", sep = ""),width=640,height=480)

mycol2 = c(rev(brewer.pal(9,"Reds")), brewer.pal(8,"Blues"))

mybreaks2= c(seq(-0.00000033,0.00000008,length.out = 18))

Longitude2 =Longitude1-360 ;  Latitude2 = rev(Latitude1)
```

```
image.plot(Longitude2,Latitude2,Okubo, breaks = mybreaks2 , col = mycol2,horizontal = T,
           legend.lab = "Okubo Wiess Parameter (1/s^2)")

title(main = paste("Okubo Wiess Parameter " ,dtime[i]) ,
      xlab = "Longitude", ylab = "Latitude")

map(database = "world", resolution = 0, add = T)

dev.off()

}



# Converting jpegs to a gif (Vorticity)

ani.options(interval = 0.25) # frame Delay
```



```
files = sprintf(paste("Vorticity_%i.jpeg"), 90:124) # choosing jpegs

im.convert(files =files, output = "Vorticity.gif") # jpegs to gif

## Executing:
## convert -loop 0 -delay 25 Vorticity_90.jpeg Vorticity_91.jpeg Vorticity_92.jpeg
##    Vorticity_93.jpeg Vorticity_94.jpeg Vorticity_95.jpeg Vorticity_96.jpeg
##    Vorticity_97.jpeg Vorticity_98.jpeg Vorticity_99.jpeg Vorticity_100.jpeg
##    Vorticity_101.jpeg Vorticity_102.jpeg Vorticity_103.jpeg Vorticity_104.jpeg
##    Vorticity_105.jpeg Vorticity_106.jpeg Vorticity_107.jpeg Vorticity_108.jpeg
##    Vorticity_109.jpeg Vorticity_110.jpeg Vorticity_111.jpeg Vorticity_112.jpeg
```

```
##    Vorticity_113.jpeg Vorticity_114.jpeg Vorticity_115.jpeg Vorticity_116.jpeg
##    Vorticity_117.jpeg Vorticity_118.jpeg Vorticity_119.jpeg Vorticity_120.jpeg
##    Vorticity_121.jpeg Vorticity_122.jpeg Vorticity_123.jpeg Vorticity_124.jpeg
##    'Vorticity.gif'
## Output at:  Vorticity.gif
```

```
# Converting jpegs to a gif (Okubo Weiss)

ani.options(interval = 0.25) # frame Delay

files = sprintf(paste("Okubo_%i.jpeg"), 90:124) # choosing jpegs

im.convert(files =files, output = "Okubo.gif") # jpegs to gif
```

```
## Executing:
## convert -loop 0 -delay 25 Okubo_90.jpeg Okubo_91.jpeg Okubo_92.jpeg
##    Okubo_93.jpeg Okubo_94.jpeg Okubo_95.jpeg Okubo_96.jpeg Okubo_97.jpeg
##    Okubo_98.jpeg Okubo_99.jpeg Okubo_100.jpeg Okubo_101.jpeg Okubo_102.jpeg
##    Okubo_103.jpeg Okubo_104.jpeg Okubo_105.jpeg Okubo_106.jpeg Okubo_107.jpeg
##    Okubo_108.jpeg Okubo_109.jpeg Okubo_110.jpeg Okubo_111.jpeg Okubo_112.jpeg
##    Okubo_113.jpeg Okubo_114.jpeg Okubo_115.jpeg Okubo_116.jpeg Okubo_117.jpeg
##    Okubo_118.jpeg Okubo_119.jpeg Okubo_120.jpeg Okubo_121.jpeg Okubo_122.jpeg
##    Okubo_123.jpeg Okubo_124.jpeg 'Okubo.gif'
## Output at:  Okubo.gif
```

```
### dplyr's storms dataset (NOAA Atlantic hurricane database (HURDAT2))

# %>% is dplyr's pipe, this pipe selects variables and
# enables to apply the following function to the selected variables

# Time variable is created with dplyr's mutate function

hurricane <- select(storms,name,year, month,day,hour, lat,long,
                    category) %>% mutate( time = paste(paste(year,month,day , sep = "/"), paste(hour, ":

# Latitude, Longitude, and Category variables are obtained with pull function
lat <- hurricane[6998:7029,] %>% pull(lat) ; Latitude =as.data.frame(lat)
lon <- hurricane[6998:7029,] %>% pull(long) ; Longitude= as.data.frame(lon)
category <- hurricane[6998:7029,] %>% pull(category)

Hurricane_Category =as.data.frame(as.numeric(as.character(category)))

katrina = data.frame(Latitude,Longitude,Hurricane_Category)

jpeg(file=paste("Track" ,".jpeg", sep = ""),width=1080,height=720)

# US map is obtained from Google Maps with get_map fuction
map = get_map(location = c(lon = -95.3632715, lat = 29.7632836), zoom = 4 ,  scale = "auto")
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=29.763284,-95.363271&zoom=4&size=
```

```
# Locations are settled with geom_point, geom_path, and aes functions

hurricane_category <- ggmap(map) +
  ggtitle(paste("                              Hurricane Katrina Track between",
                katrina_time[1],"and",katrina_time[32] ,sep = " ")) +
```

```
  geom_point(aes(x=Longitude, y=Latitude, size= Hurricane_Category)
             ,data = katrina, alpha= 0.5, color ="darkred") +
  geom_path(aes(x=Longitude, y=Latitude) ,data = Hurricane_Category, alpha=1)

hurricane_category
```

```
## Don't know how to automatically pick scale for object of type data.frame.  Defaulting to
continuous.
```

```
dev.off()

## RStudioGD
##         2
```

The R session information (including the OS info, R version and all packages used):

```
sessionInfo()

## R version 3.4.2 (2017-09-28)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Sierra 10.12.6
##
## Matrix products: default
## BLAS: /System/Library/Frameworks/Accelerate.framework/Versions/A/Frameworks/vecLib.framework/Versions
## LAPACK: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] rmarkdown_1.8      knitr_1.18         bindrcpp_0.2       ggmap_2.6.1
##  [5] ggplot2_2.2.1      dplyr_0.7.4        animation_2.5      fields_9.0
##  [9] maps_3.2.0         spam_2.1-1         dotCall64_0.9-04   RColorBrewer_1.1-2
## [13] ncdf4_1.16
##
## loaded via a namespace (and not attached):
##  [1] reshape2_1.4.3    lattice_0.20-35   colorspace_1.3-2  htmltools_0.3.6
##  [5] yaml_2.1.16       utf8_1.1.3        rlang_0.1.6       pillar_1.0.1
##  [9] glue_1.2.0        sp_1.2-5          jpeg_0.1-8        bindr_0.1
## [13] plyr_1.8.4        stringr_1.2.0     munsell_0.4.3     gtable_0.2.0
## [17] RgoogleMaps_1.4.1 mapproj_1.2-5     evaluate_0.10.1   labeling_0.3
## [21] highr_0.6         proto_1.0.0       Rcpp_0.12.14      geosphere_1.5-7
## [25] scales_0.5.0      backports_1.1.2   rjson_0.2.15      png_0.1-7
## [29] digest_0.6.13     stringi_1.1.6     rprojroot_1.3-2   cli_1.0.0
## [33] tools_3.4.2       magrittr_1.5      lazyeval_0.2.1    tibble_1.4.1
## [37] crayon_1.3.4      pkgconfig_2.0.1   assertthat_0.2.0  rstudioapi_0.7
## [41] R6_2.2.2          compiler_3.4.2

Sys.time()

## [1] "2018-01-11 01:21:06 +03"
```