R y radares: muchos datos y algunos resultados

```
#Primero  lo primero, cargamos las librerias que vamos a necesitar

library(ncdf4) #Permite leer archivos con formato .nc
library(ggplot2) #Gráficos!
library(lubridate) #Permite trabajar con fechas de manera sencilla
library(stringi) #Permite trabajar con strings
library(reshape2) #Para usar la función melt
library(scales) #Para escalas de colores que usa ggplot
```

En Ciencias de la Atmósfera trabajamos con muchos datos. Esos datos pueden ser observaciones, salidas de modelos meteorológicos o reanálisis.

En particular yo trabajo con datos de radares meteorológicos. El radar emite una señal que rebota en las nubes o en lo que haya en la atmósfera y vuelve para ser recibida por el radar. Con esto sensan la presencia de nubes, el tamaño y la forma de gotas y en algunos casos es posible medir viento.

El radar gira sobre su eje enviando pulsos en cada ángulo horizontal o azimut. Al terminar una vuelta, se inclina hacia arriba y cambia su ángulo de elevación para volver a empezar la siguiente vuelta. Más o menos como muestran en este video. El pulso de radar llega hasta cierta distancia (rango) y el radar es capaz de medir a que distancia se encuentran las cosas calculando cuanto tiempo tardar en en llegar la señal.

En pocos minutos el radar escanea todo (o casi) el volumen de atmósfera que tiene alrededor, adquiriendo muchos datos con una gran resolución espacial y temporal.

Por ejemplo el radar del INTA que se encuentrá en Paraná escanea:

12 elevaciones x 360 azimuts x 480 gates* = 2073600 datos cada 10 minutos!

*Gates refiere a la cantidad de puntos de medición entre el radar y la distancia máxima. A veces se le llama rangos. Por ejemplo, este radar puede observar cosas que estan en un radio de 240km y con datos cada 500m.

¿Cómo hago para trabajar con un volumen de datos?

Los datos de radar vienen en algún formato mas o menos standar que depende del software pero hay maneras de convertirlo a un tipo de archivo bastante común en nuestra diciplina, el netcdf.

Este archivo tiene muchísima información organizada en cierta estructura con variables y su información asociada.

Lo bueno es que R tiene una librería para leer este tipo de archivos. La librería se llama *ncdf4*.

```
radar <- nc_open("Datos/cfrad.20160114_060005.000_to_20160114_060431.001_INTA_Parana_SUR.nc")
azimut <- ncvar_get(radar, 'azimuth')   #"Saca" la variable llamada azimuth
elevacion <- ncvar_get(radar, 'elevation') #"Saca" la variable llamada elevation
rango <- ncvar_get(radar, 'range')      #"Saca" la variable llamada range
V <- ncvar_get(radar, 'Vda')            #"Saca" la variable viento

V[V > 99999] <- NA                      #Trampa para definir datos faltantes en esta variable
```

La variable que me interesa es **V**, que corresponde al viento que mide el radar. Los datos estan organizados en una matriz donde las filas (480) corresponden al rango (o sea la distancia del dato medido a la ubicación del radar) y las columnas corresponden al momento en que se tomó ese dato contado desde el inicio del escaneo (4327). El dato del tiempo puede ser traducido al azimuth (o sea el ángulo horizontal) ya que sabemos que ángulo midió en el momento cero y por lo tanto cada columna de V se va a corresponder con un elemento del vector azimut.

Por otro lado extraje la variable **rango**, que me da esa grilla espacial. Es un vector con 480 elementos. También tenemos **elevación** y **azimut**, son dos vectores del mismo tamaño 4327 elementos. En el primero el valor de cada ángulo de elevación se repite tantas veces como cantidad de ángulos de azimut se miden. El vector azimut contiene el valor de cada ángulo de azimuth medido para cada elevación.

¿Cómo cuernos trabajamos con esta información?

Mi objetivo es analizar la variable viento, cómo se comporta en distintos niveles, cómo cambia con la distancia al radar, etc. En ese proceso podemos hacer cálculos estadísticos, hacer gráficos, etc. Entonces necesitamos organizar los datos de tal manera que sean útiles. ¿Cómo? En un *dataframe*.

Un dataframe permite ordenar los datos de tal manera que cada fila corresponda a una observación (en este caso el valor del viento en un punto) y acompañando al dato, las variables (la posición de ese dato en el espacio -rango, azimuth, elevación-, podría ser el tiempo también). Entoces quiero un data frame con V, azimuth, rango y elevación en columnas.

Podría haber elegido ordenar los datos en distintas matrices según el ángulo de elevación y tal vez armar un array multidimensional (algo así com una matriz pero con 3 dimensiones). Y seguro hay muchas otras maneras de representar los datos. Pero el formato que elegí, *tidy data* es el que mejor se lleva con ggplot.

En primero lugar le asigno nombre a las filas y columnas de la matriz V. Recordemos que las filas se corresponden con los valores del rango y las columnas con los valores del azimut.

```
dimnames(V) <- list(rango = rango, azimut = azimut) #Asigno valores a las filas y columnas

V[1:10,1:4] #Muestra las primeras filas y columnas de V
```

```
##       azimut
## rango  214.024658203125 215.0244140625 216.018676757812 217.018432617188
##   250                NA             NA                NA                NA
##   750                NA             NA                NA                NA
##   1250       -4.1889763     -0.1570866                NA                NA
##   1750        0.4188976      0.4188976       -0.3665354                NA
##   2250       -4.5555115     -4.3984251       -5.8645668        -6.335827
##   2750       -0.9948819     -2.9846456        1.8326771        -3.717716
##   3250       -3.4035432     -3.7177165       -2.9322834        -1.937402
##   3750       -6.2834644     -3.5082676       -3.5082676        -6.964173
##   4250       -2.8799212     -0.4188976       -1.4137795        -0.261811
##   4750       -3.7177165     -4.0842519       -2.8275590        -2.932283
```

Si observamos la matriz, veremos que cada observación esta asociada a un rango y un azimut determinado.

Para armar el dataframe tengo que "desarmar" la matriz V de tal manera que me quede una observación por fila. Una opción es usar la función `melt()` que toma cada columna de V y la ordena en una única columna larga con estructura de dataframe con la información del rango y el azimut incorporada.

```
vol <- melt(V, value.name = "V") #Crea el dataframe vol con las observaciones de V
```

Sólo nos falta la columna elevación que no estaba asociada a la matriz V. Ahora usamos la función `rep()` que repite los valores del vector elevación la cantidad de veces necesarias para completar el dataframe y que cada observación tenga su altura correspondiente.

```
vol$elevacion <- rep(elevacion, each = length(rango))

head(vol) #Muestra las primeras filas del dataframe, notar que si hacemos head(V) el resultado no es el
```

```
##   rango   azimut          V elevacion
## 1   250 214.0247         NA       0.5
## 2   750 214.0247         NA       0.5
## 3  1250 214.0247 -4.1889763       0.5
```

```
## 4  1750 214.0247  0.4188976       0.5
## 5  2250 214.0247 -4.5555115       0.5
## 6  2750 214.0247 -0.9948819       0.5
```

Listo, tenemos nuestro dataframe. Podríamos hacer un gráfico para ver que pinta tiene.

Los gráficos mas usados muestran la información para un determinado ángulo de elevación. Por lo tanto voy a necesitar separar los datos del dataframe para algún ángulo.

```
elev <- unique(elevacion) #Genera un vector con los ángulos existentes. Esto me permitirá hacer un mejo
```

Abajo van a encontrar el código para crear un gráfico de esta pinta. Les recomiendo ir comentando con # las distintas líneas para ver el resultado.

Como ven dentro de la función `ggplot()` indicamos que queremos usar los datos del dataframe vol y en particular sólo los que cumplen con la condición de que la elevación sea la número 3 (1.3) y que no sean NA (o sea sin dato válido o faltante). Dentro del aes indico que en el eje x quiero el azimut y en el eje y quiero el rango (dividio por 1000 para tener los datos en kilómetros).

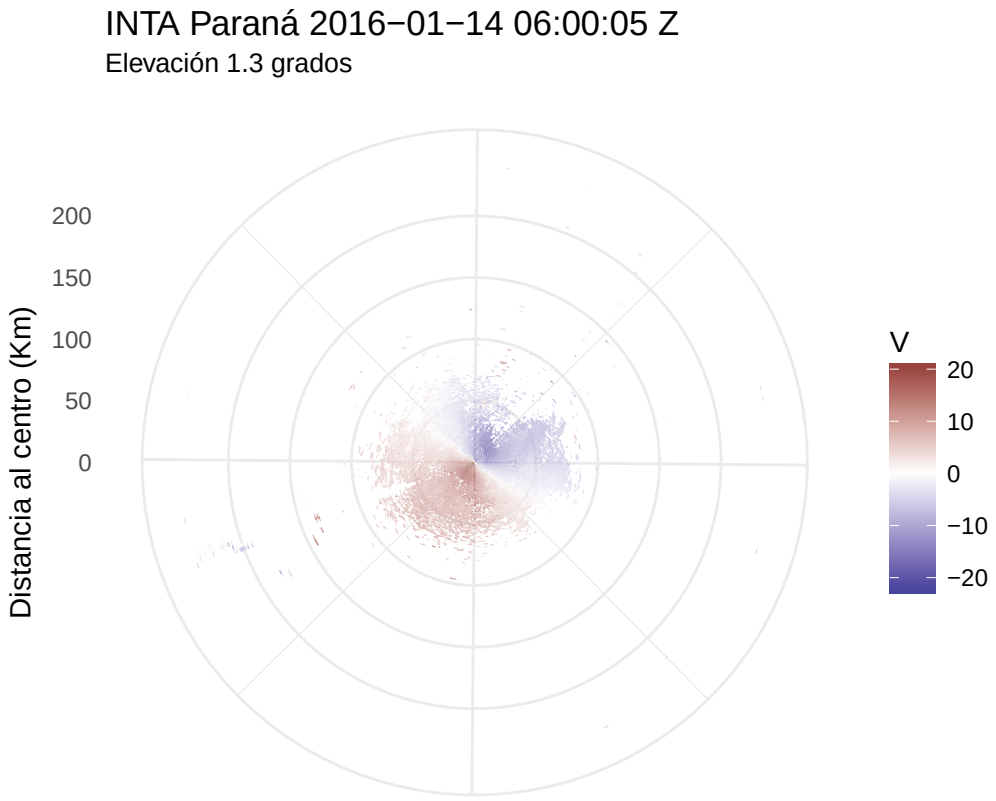
Elijo el `geom_tile()`, que grafica rectangulos de tal manera que todo el espacio a graficar queda cubierto. En el aes del geom indico la variable a graficar V y al hacer esto, el color indicará la magnitud del viento.

Lo que sigue son algunos detalles estéticos, la escala de colores (un detalle muy importante a la hora de hacer un gráfico pero fuera de tema), nombres de ejes, título y el tema.

Pero en el medio hay una función muy importante, la proyección. Normalmente vemos gráficos en coordenadas cartesianas, 2 ejes perpendiculares que solemos llamar x e y. Pero en este caso la geometría de los datos no está ordenada de esa manera. Los datos dependen de la distancia al centro (dirección radial) y del ángulo azimutal y por lo tanto tiene mas sentido graficarlos en coordenadas polares.

Un problema importante es que ggplot es muy lento para traducir los datos y generar el gráfico en coordenadas polares y en particular el geom_tile es es muy ineficiente. Les propongo comentar esa línea y ver la diferencia en el tiempo. También van a ver porque es tán importante la elección del tipo de proyección.

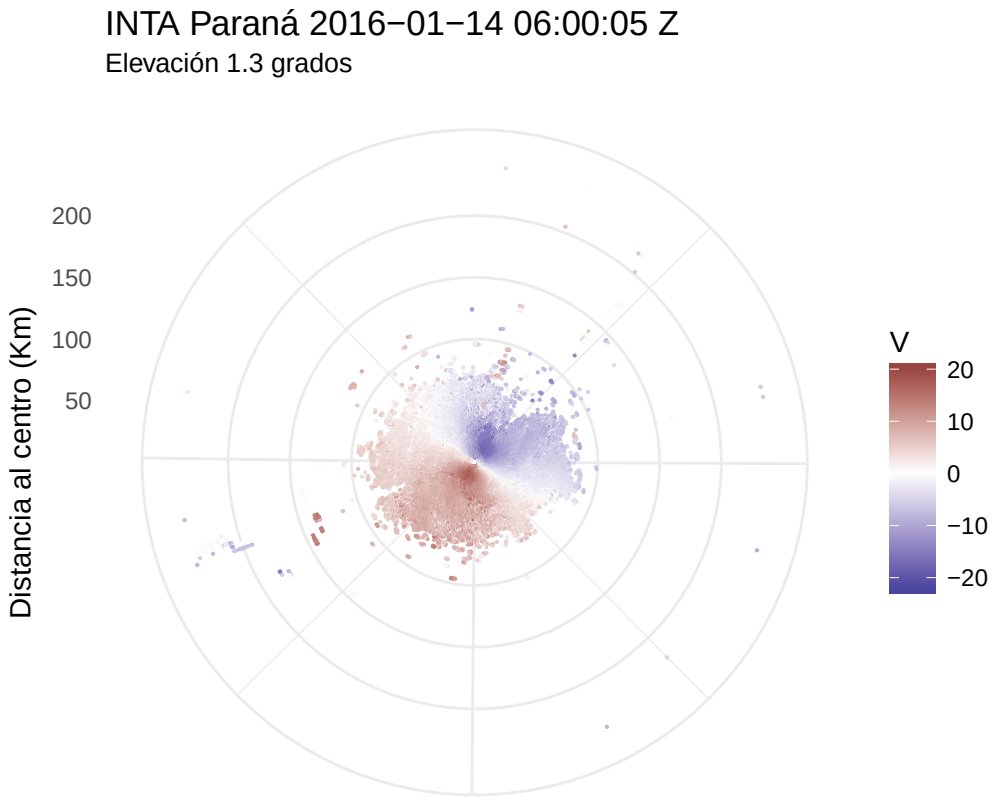
```
#Ojo este chunk demora mucho, no se asusten!
ggplot(subset(vol, elevacion == elev[3] & !is.na(V)), aes(azimut, rango/1000)) +
  geom_tile(aes(fill = V)) +
  scale_fill_gradient2(low = muted("blue"), high = muted("red")) +
  scale_x_continuous(name = NULL, labels = NULL, breaks = seq(0, 359, 90)) +
  ylab("Distancia al centro (Km)") +
  labs(title = "INTA Paraná 2016-01-14 06:00:05 Z", subtitle = "Elevación 1.3 grados") +
  coord_polar() +
  theme_minimal()
```



Antes de seguir analizando el código veamos que nos muestra el gráfico. Estamos viendo el viento que mide el radar cuando da una vuelta completa a un determinado ángulo de elevación, en este caso 1.3. Pero el viento que mide el radar en realidad no es el viento real, es sólo la parte del viento que va en la dirección paralela a rayo que emite. Entonces, en el gráfico los tonos rojizos correspode a viento que va hacia el radar y los tonos azules es viento que se va del radar. El color blanco representa viento nulo, al menos en la dirección paralela al rayo. A simple vista podemos ver que en esa región tenemos viento que sopla del suroeste (SO). Hay muchas otras características que se pueden analizar en este tipo de gráficos pero lo dejamos la pa próxima o las mas curiosas.

La elección de geom_tile no es arbitraria y pese a ser poco eficiente es preferible a otros geoms. Por ejemplo `geom_point()` se grafica mucho más rapidamamente pero los puntos se solapan y es necesario modificar su tamaño con el argumento size. Además agrego la función `scale_size_area` que permite modificar el área de los puntos, en este caso el pongo un límite máximo porque quiero que los puntos sean chiquitos porque tengo muchísimos puntos y no quiero que se superpongan. Con esto obtenemos un gráfico bastante similar al anterior pero en mucho menos tiempo. Les propongo comentar la linea que dice scale_size_area y graficar nuevamente para ver la diferencia.

```
ggplot(subset(vol, elevacion == elev[3] & !is.na(V)), aes(azimut, rango/1000)) +
  geom_point(aes(color = V, size = rango)) +
  scale_color_gradient2(low = muted("blue"), high = muted("red")) +
  scale_size_area(max_size = 0.1) +
  scale_x_continuous(name = NULL, labels = NULL, breaks = seq(0, 359, 90)) +
  guides(size = "none") +
  ylab("Distancia al centro (Km)") +
  labs(title = "INTA Paraná 2016-01-14 06:00:05 Z", subtitle = "Elevación 1.3 grados") +
  coord_polar() +
  theme_minimal()
```



Uno de los objetivos de mi tesis es estudiar y caracterizar la Capa Límite Atmosférica (que es mas o menos los primeros 2000 metros desde el suelo) a partir de los datos de viento del radar. En particular quiero ver como evoluciona a lo largo del día, y para eso necesito los volumenes medidos por el radar durante 24 horas, o sea entre 120 y 144 volumenes de datos.

Como vimos mas arriba el radar mide el viento en la dirección paralela al rayo que emite, pero para el análisis de la Capa Límite necesito el viento real (horizontal). Por suerte hay algorithmos que permiten transformar ese viento *radial* en el viento real basados en VAD. Lamentablemente no hay librerías para esto en R pero existen paquetes escritos en otros idiomas que se pueden utilizar.

Particularmente yo escribí mi propio algoritmo en python principalmente porque necesitaba que fuera mas customizable que los que uno encuentra en la bibliografía (si alguna le interesa python puedo compartirles el código). Este código genera un archivo .csv con la información del viento *promedio* en la región del radar (tanto la magnitud como su dirección) y su variación con la altura, junto con otras variables como la estimación del error al estimar el viento y la cantidad de datos utilizados para esto.

Con esta información cada 10 minutos (o sea entre 120 y 144 archivos para cada día) me propongo analizar las características de la Capa Límite. Voy a mostrarles 2 gráficos que resumen las características mas importantes.

El siguiente código me permite leer todos los archivos juntos y armar el dataframe. En este caso no necesito ordenar la información ya que los .csv los genero yo de acuerdo a lo que necesito.

```
filename <- (Sys.glob("Datos/20160114_240/vda*")) #Lee todos los archivos para el 14/01/2016
datetime <- ymd_hms(stri_sub(filename, from = 23, to = 43)) #Extrae la hora y el día
#de cada archivo

#Loop para leer cada archivo y agregarlo al final del dataframe "vad_14_240"
for (i in 1:length(filename)){
  temp <- read.csv(filename[i], sep = ";", dec = ".", na.strings = "-9999")
  date_time <- datetime[i]
  temp$date_time <- date_time
```

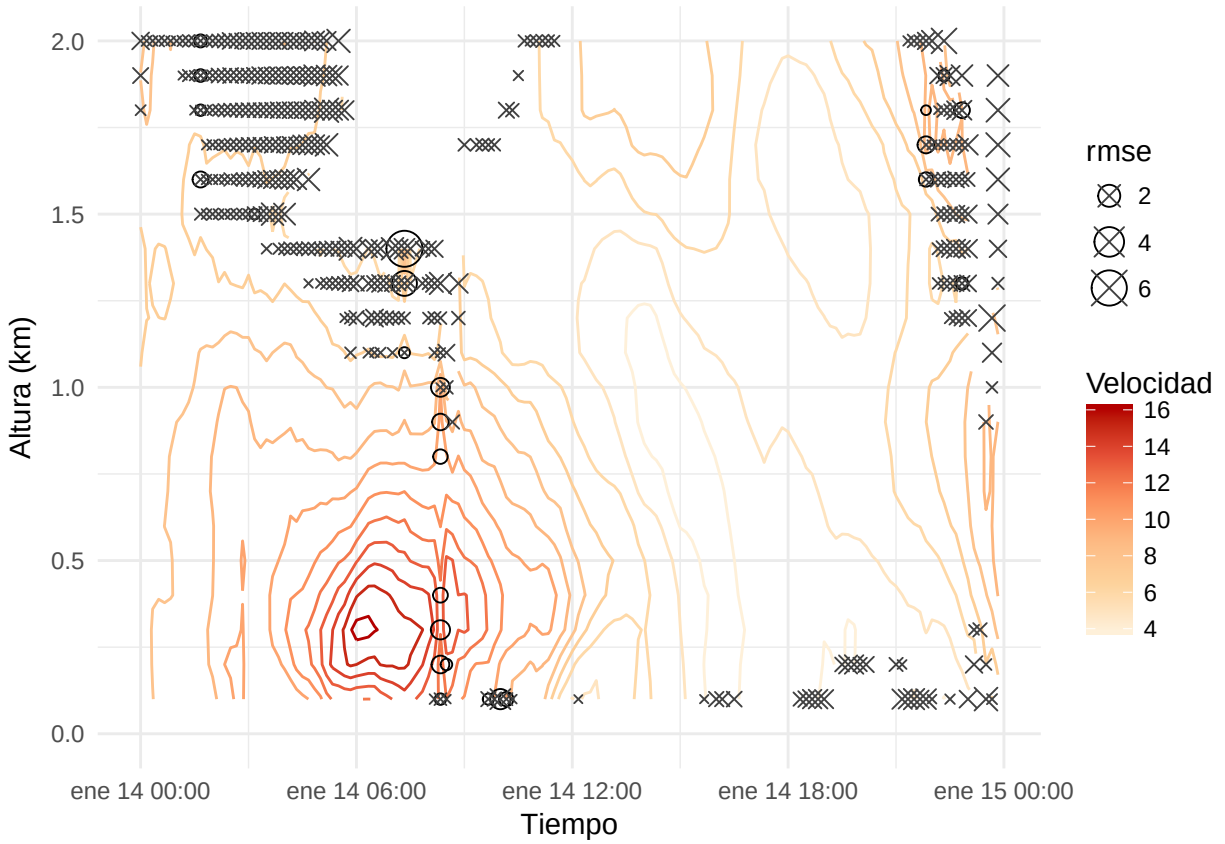
```
  if (i == 1){
    vad_14_240 <- temp
  } else {
    vad_14_240 <- rbind(vad_14_240, temp)
  }
}
```

El primer gráfico nos permitirá analizar la variación de la magnitud del viento (variable spd) a lo largo del día (date_time) y para cada altura (ht).

El geom elejido es contour, que grafica contornos del mismo valor de viento y nos permite determinar donde el viento es máximo (colores rojos) y donde es mínimo (naranja claro). Una de las características mas importantes es ese máximo que se ve a bajo a la izquierda, está a unos 300 metros de altura y ocurre en horas nocturas. Ese máximo se llama jet de capaz bajas.

Pero van a ver que agrego otros geoms al gráfico. Son geom_point pero cambiando la forma con el argumento shape. El número 1 corresponde a circulos no llenos y el 4 son las cruces. Esos geoms grafican el error (rmse) cuando es mayor a 0.5 y el tamaño del punto depende de la magnitud del error. Hay dos tipos de errores ya que uno permite ver que tan dispersos son los datos respecto de su valor medio y el otro mide el error en la estimación del viento.

```
ggplot(vad_14_240, aes(date_time, ht)) +
  geom_contour(aes(z = spd, color = ..level..), binwidth = 1) +
  scale_color_distiller(name = "Velocidad", type = "seq", palette = 8, direction = 1) +
  geom_point(data = subset(vad_14_240, rmse > 0.5), aes(size = rmse), shape = 1, color = "black") +
  geom_point(data = subset(vad_14_240, rmse2 > 0.5), aes(size = rmse2), shape = 4, color = "grey25") +
  ylim(c(0,2)) +
  xlab("Tiempo") +
  ylab("Altura (km)") +
  theme_minimal()
```



Con el segundo gráfico queremos ver como cambia la dirección del viento a lo largo del día. Pero ahora la variable no es continua como si lo es la magnitud del viento porque está expresada en ángulos. En meteorología se suelen usar (barbas)[https://content.meteoblue.com/es/ayuda/variables-meteorologicas/viento] que son símbolos que resumen la dirección y la magnitud del viento. Lamentablemente -por ahora- no se pueden graficar en R.

Una posible solución es usar flechas graficadas con `geom_text()`. Acá la cosa se pone un poco mas compleja, pero en resumidas palabras el `geom_arrow()` que ven en el código es una función armada por fuera del paquete ggplot que usa geom_text y grafica flechas en la dirección que corresponde al ángulo.

Como tengo muchos datos y las flechas son grandes, sólo voy a graficar los datos que corresponden a las horas redodas, o sea los datos de las 13 pero no los de las 13:10 y por eso el subset.

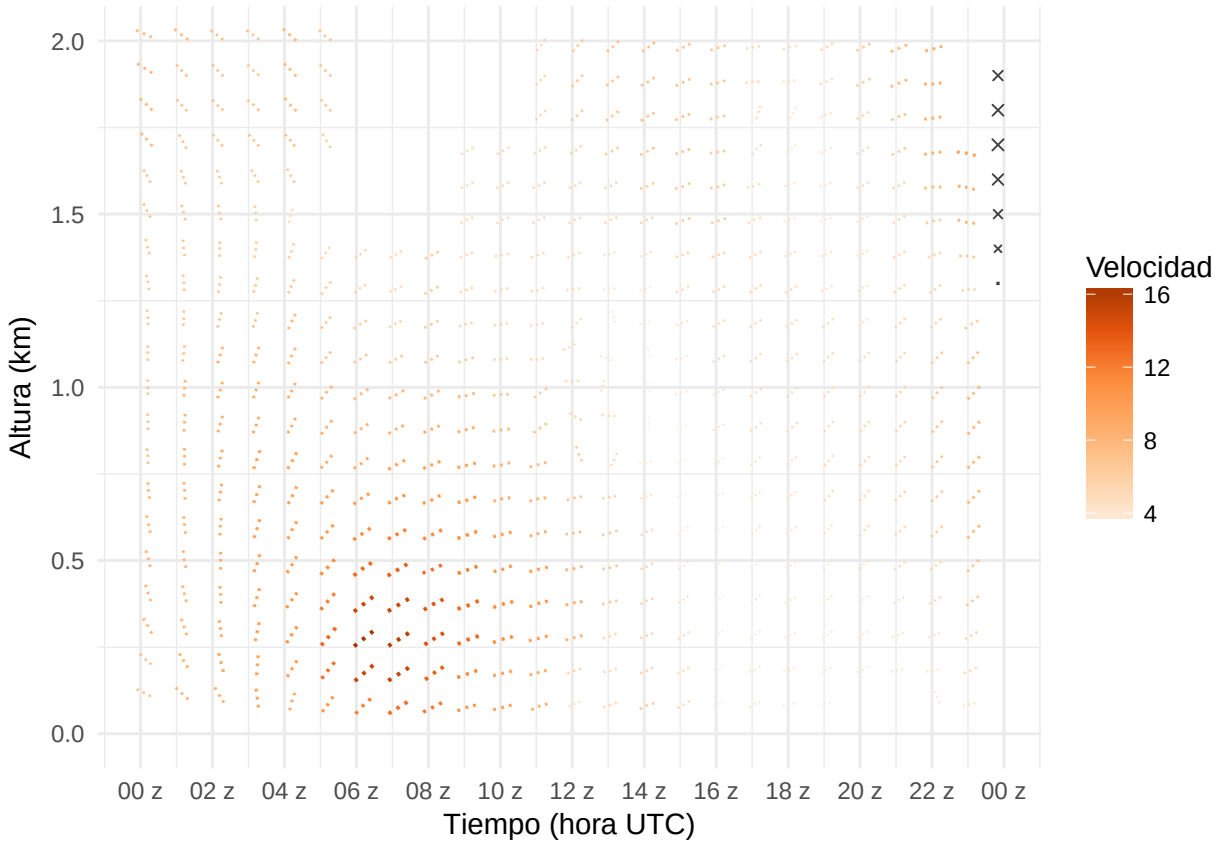
De nuevo grafico los errores y el resto del código tiene que ver con hacer que las cosas se vean bien. Por ejemplo el argumento range en scale_size me da el tamaño mínumo y máximo de las fechas, en scale_color indico la escala de colores y el nombre para la leyenda.

La función Scale_x_datetime me permite indicar la leyenda del eje x. Cómo se trata de datos temporales tienen un formato distinto. Si alguna vez tienen que trabajar con este tipo de datos la librería *lubridate* es una muy buena herramienta.

```
geom_arrow <- function(aes, direction = 1, start = 0, ...) {
    # geom para graficar flechas más fácilmente.
    # aes requeridos :
    #  * mag (magnitud) y angle (ángulo en radianes)
    #  * vx (velocidad en dirección x) y vy (velocidad en dirección y)
    #
    # Otros parámetros:
    #   direction: dirección del ángulo. 1 para antihorario, -1 para horario
```

```
    #   start: ángulo de inicio. 0 para empezar desde el eje x, -1/4*pi para
    #   el ángulo meteorológico
    #   ... : otros parámetros para geom_text
    if (!is.null(aes$angle) & !is.null(aes$mag)) {
        angle <- deparse(aes$angle)
        aes.angle <- paste0(start,  "+", direction, "*", angle)
        geom_text(aes_string(size = aes$mag,
                             angle = aes.angle,
                             color = aes$colour),
                  label = "\u27f6", ...)
    } else if (!is.null(aes$vy) & !is.null(aes$vx)) {
        aes.angle <- paste0("atan2(", aes$vy, ", ", aes$vx, ")*180/pi")
        aes.size <- paste0("sqrt(", aes$vx, "^2 + ", aes$vy, "^2)")
        geom_text(aes_string(size = aes.size, angle = aes.angle,
                             color = aes$colour), label = "\u27f6", ...)
    } else {
        stop("geom_arrow needs either angle and mag or vx and vy")
    }
}
```

```
ggplot(subset(vad_14_240, minute(date_time) == 0), aes(date_time, ht)) +
  geom_arrow(aes(mag = spd, angle = di, color = spd)) +
  geom_point(data = subset(temp, rmse > 0.5), aes(size = rmse), shape = 1, color = "black") +
  geom_point(data = subset(temp, rmse2 > 0.5), aes(size = rmse2), shape = 4, color = "grey25") +
  scale_size_continuous(range = c(0, 5), guide = "none") +
  scale_color_distiller(palette = "Oranges", name="Velocidad", direction = 1) +
  xlab("Tiempo (hora UTC)") + ylab("Altura (km)") +
  ylim(c(0,2)) +
  scale_x_datetime(date_breaks = "2 hour", date_labels = "%H z") +
  theme_minimal()
```



Finalmente con este gráfico podemos ver como va cambiando de dirección el viento a lo largo del día y también como cambia con la altura para un mismo momento.

Con suerte estos gráficos me van a permitir sacar algunas conclusiones, proximamente en mi tesis!

Datos de contacto

paobcorrales@gmail.com

/paobcorrales

O podés crear un issue en mi repositorio si te animás a GitHub!