Clustering de patrones climáticos para ciclos productivos de cultivos

Centro Internacional de Agricultura Trópical grupo de Big-Data AEPS Hugo Dorado y Andrés Aguilar

November 10, 2016

Introducción

El script `Procesamiento_Clima.R`, contiene funciones que permiten: procesar, evaluar y agrupar patrones climáticos de eventos productivos de cultivos, teniendo en cuenta su correspondiente serie climática a nivel diario, desde la siembra hasta la cosecha. El propósito es identificar y agrupar periodos que tengan semejanzas en tendencias y características climáticas (homólogos); con el fin de determinar entre ellos diversas prácticas tales como: variedades sembradas o fertilizaciones aplicadas, que hayan tenido buenos resultados y sirvan de información relevante para tomar mejores decisiones en el momento en el que ocurra un patrón similar.

El proceso comienza con la lectura de la base de datos de clima, proveniente de una estación meteorológica y una base de datos de cultivo que contiene las variables necesarias para proceder con el análisis, luego se hace una vinculación de cada evento de producción con su historial climático; para después realizar un análisis de clustering jerárquico definiendo a dtw como la distancia y por último una descripción de los clustering conformados de acuerdo al rendimiento y fechas de siembra.

Cargar librerías en R y preparación de datos

En el siguiente comando se cargan las librerías necesarias para operar las funciones que se utilizaran en el script, en caso de que alguna falte, desde R se deberá instalar utilizando el comando `install.packages`. Una vez estemos seguros que estén disponibles para ser cargadas, es posible proceder con el codigo.

```
libs= c("reshape2","dtwclust","dtw","gtools","agricolae")

lapply(libs, require, character.only=T)
```

El siguiente paso consiste en definir la ubicación del directorio de trabajo donde se encuentran los datos climáticos y de manejo; en este caso asumimos que ambos se encuentran dentro de la misma carpeta.

Dentro del comando `source` se indicará la ubicación del archivo `funciones_cluster_temporal.R` donde se encuentran las funciones de clustering.

```
setwd("D:/GIT_HUB_REPOSITORIOS/CLUSTERING_PATRONES_CLIMA")

source('funciones_cluster_temporal.R')
```

A continuación se lee la base de datos de eventos productivos de la cual se espera que en cada fila se encuentre un ciclo de siembra; está matriz deberá contener como mínimo: fecha de siembra, fecha de cosecha y el rendimiento. También es relevante que tenga un ID único por fila. De resto, puede contener sin inconvenientes variables adicionales. En este punto es muy importante tomar en cuenta dos cosas: (1) el nombre asignado a las fechas y rendimiento, y (2) El formato en el que R ha leido la fecha.

Por otro lado también se espera que los datos en este punto ya se hayan limpiado, es decir, que se hayan validado los valores para rendimientos en las unidades correspondientes y además que los días entre fecha y cosecha se mantengan dentro de un intervalo de días posible para un cultivo.

```
base_eventos <- read.csv("DATOS/maiz_cluster_cerete.csv",row.names = 1)

head(base_eventos)
```

```
##      ID_LOTE ID_FINCA FECHA_SIEMBRA TIPO_CULTIVO FECHA_COSECHA Rendimiento
## 2224    1984     1879     5/30/2015         Maiz     9/14/2015        4000
## 2276    2104     1985     5/15/2015         Maiz      9/2/2015        3500
## 2225    1983     1880     5/30/2015         Maiz     9/19/2015        2130
## 804      680      674     9/29/2013         Maiz     1/20/2014        6800
## 2042    1826     1726      5/3/2015         Maiz     8/25/2015        2600
## 2282    2101     1988     5/23/2015         Maiz     9/14/2015        4000
```

Cómo podemos observar en nuestro ejemplo se lee desde R la fecha en una forma que va en orden: mes, día y año; además está separada por `/`, posteriormente indicamos que R reconozca estas variables con el formato apropiado.

```
base_eventos$FECHA_SIEMBRA <- as.Date(base_eventos$FECHA_SIEMBRA,"%m/%d/%Y")

base_eventos$FECHA_COSECHA <- as.Date(base_eventos$FECHA_COSECHA,"%m/%d/%Y")
```

A continuación procedemos a leer los archivos climáticos, se supone que deben estar en escala diaria y se debe haber efectuado métodos de control de calidad; en caso de que aún no lo hayamos hecho podemos tomar como referencia los métodos propuestos aquí.

De acuerdo a los archivos que tengamos de clima, existen dos posibilidad de lectura, dependiendo de la manera en que vengan los archivos: (1) cada variable separada en un archivo .txt o (2) las 5 variables ya consolidadas en un csv.

(1) Cada variable separada en un archivo .txt

En esta posibilidad es necesario reconocer el orden en que R lee los archivos climáticos, esto lo podemos identificar con la función `list.files`, después es necesario establecer el nombre con el que llamaremos a cada una de las variables; en nuestro caso les hemos puesto: `ESOL, RAIN, RHUM, TMAX Y TMIN`; note que el orden que aparece en `list.files` fue conservado. En todos los archivos, la fecha debe tener el mismo formato y no deben faltar días.

```
list.files("DATOS/DIVIDIDOS")
```

```
## [1] "13075030_ESOL_FUS.txt" "13075030_RAIN_FUS.txt" "13075030_RHUM_FUS.txt"
## [4] "13075030_TMAX_FUS.txt" "13075030_TMIN_FUS.txt"
```

```
nombresClima <- c("ESOL","RAIN","RHUM","TMAX","TMIN")

listDatosClimaticos <-
  lapply(list.files("DATOS/DIVIDIDOS",full.names = T),read.table,header=T)

head(listDatosClimaticos[[1]])
```

```
##        Dates    Value Modif
## 1 2005-01-01 497.0498    NO
## 2 2005-01-02 500.8255    NO
```

```
## 3 2005-01-03 488.0942    NO
## 4 2005-01-04 462.1190    NO
## 5 2005-01-05 479.2272    NO
## 6 2005-01-06 360.2947    NO
```

La función `unfiDatos` permitirá unir los archivos de cada variable en un solo data frame.

```
DatosClimaticos <- unifDatos(listDatosClimaticos,nombresClima)

DatosClimaticos$DATE <- as.Date(DatosClimaticos$DATE)
```

(2) Todas las variables en un solo archivo

Existe también la posibilidad de que las variables ya estén en una sola matriz, dado ese caso solo debemos actualizar el formato de la fecha, nuevamente debemos estar atentos a los encabezados que representan cada una de las variables.

```
#Unidos

DatosClimaticos <- read.csv("DATOS/13075030_JOINT_CERETE.csv")

#Acomodar formato de fecha

head(DatosClimaticos)
```

```
##       DATE RAIN TMAX     TMIN RHUM     ESOL
## 1 1/1/2005    0 33.2 23.44283   79 497.0498
## 2 1/2/2005    0 33.6 22.83606   79 500.8255
## 3 1/3/2005    0 34.4 22.85832   76 488.0942
## 4 1/4/2005    0 33.2 22.66568   78 462.1190
## 5 1/5/2005    0 33.6 23.05600   78 479.2272
## 6 1/6/2005    0 33.2 22.70614   77 360.2947
```

```
DatosClimaticos$DATE <- as.Date(DatosClimaticos$DATE,"%m/%d/%Y")
```

Vinculación de los eventos de cultivo y clima en escala diaria

En este paso se mezclan las fechas de siembra y cosecha, para construir la historia climática de cada evento, la función `joinEventsClim`, recibe como argumentos los datos de la estación climática, la base de eventos de cosecha y el nombre en que cada archivo contiene las variables de fechas.

```
ClimaEventos <-
joinEventsClim(climStat = DatosClimaticos,cropData= base_eventos,datCS = "DATE",
               sowDat = "FECHA_SIEMBRA",harvDat = "FECHA_COSECHA")

head(ClimaEventos)
```

```
##   EVENT       DATE RAIN TMAX TMIN RHUM     ESOL
## 1  2224 2015-05-30  0.4 32.8 24.6   83 273.7463
## 2  2224 2015-05-31  0.4 34.4 24.9   80 436.1400
## 3  2224 2015-06-01  0.0 32.8 25.4   82 358.0283
```

```
## 4  2224 2015-06-02  1.5 34.8 25.8   79 381.5952
## 5  2224 2015-06-03  0.0 33.6 23.5   81 425.3109
## 6  2224 2015-06-04  1.2 33.6 24.6   83 378.2936
```

`datCS` se refiere al nombre de la fecha en la estación climática, `sowDat` y `harvDat` representan los nombres de la fecha de siembra y cosecha respectivamente en la base de datos de eventos de cosecha.

Transformación de las variables

En algunas ocasiones como parte del pre procesamiento de las variables, es mejor si se realizan transformaciones; por lo menos en nuestro ejemplo no se evalúa directamente la precipitación, si no que utilizamos la escala logarítmica en base 10; para ello los días que tienen precipitación cero son remplazados con 0.05.

De tal manera que en este espacio del script se pueden aplicar todas las transformaciones que requieran las variables.

Por último creamos un vector llamado `varI`con los nombres que llevan las variables climáticas.

```
ClimaEventosTransf <- ClimaEventos

ClimaEventosTransf$LOG10RAIN <- ClimaEventosTransf$RAIN

ClimaEventosTransf$LOG10RAIN[ClimaEventosTransf$LOG10RAIN==0] <- 0.05

ClimaEventosTransf$LOG10RAIN <- log10(ClimaEventosTransf$LOG10RAIN)

varsI <- c("ESOL","LOG10RAIN","RHUM", "TMAX","TMIN")
```

Una vez se encuentra procesada la base de datos, se aplica sobre el conjunto la función `procesData`, que permitirá estandarizar las variables en valores de 0 ó 1, este paso es de suma importancia para proceder con el análisis, la función generará un archivo en formato `RData`; que luego es cargado en R a través de la función `load`. Por último el contenido es convertido en formato de serie de tiempo.

```
#Listas de eventos climaticos normalizados

procesData(climEvent = ClimaEventosTransf,idVar = "EVENT",NormMethod = 2,
           vars = varsI)

load("listClimatEvent.RData")

#Se convierte en series de tiempo multivariadas

tsnleventsN <- lapply(evenN,ts)
```

Análisis Cluster

El análisis clúster permitirá agrupar cada patrón de clima proveniente de los eventos de cosecha, para ello se utiliza la distancia dtw, que permite trabajar con series multivariadas de distintas dimensiones.

Se ha programado la función `distDtwMV` para estimar la matriz de distancia, también la función `hirarCluster` con la que se construye un clúster jerárquico.

Al construir la matriz de distancia la ejecución nos mostrará una barra de progreso que al completar el 100% nos indicará que el proceso ha terminado.

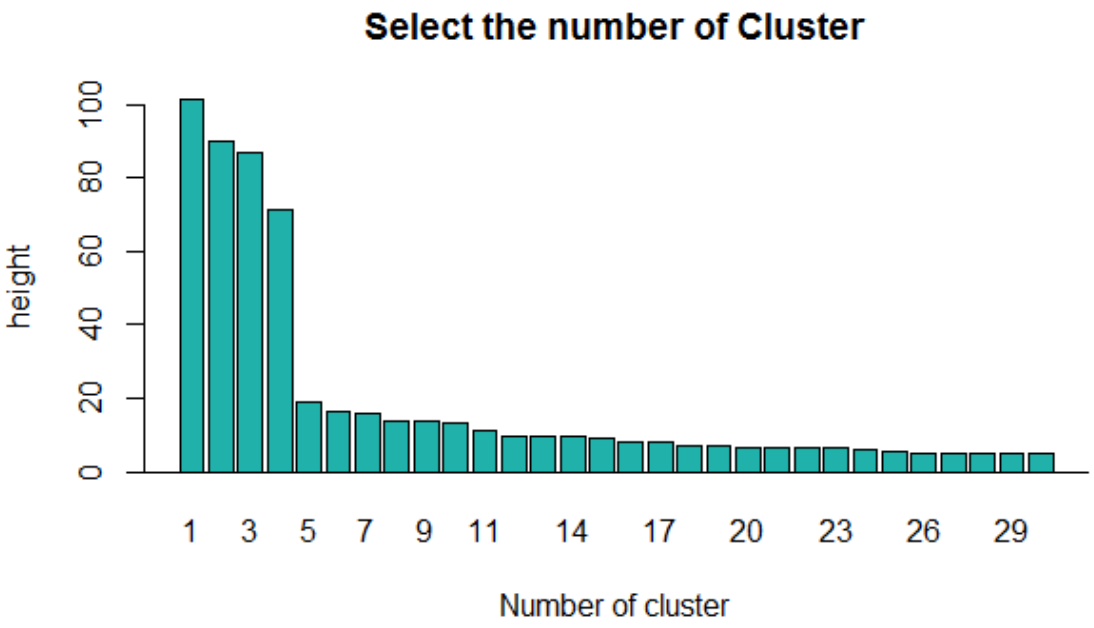
```
distAllMatrix <- distDtwMV(tsnleventsN)
```

Despues podemos ejecutar el comando `hirarCluster`, y lo primero que se debe decidir es la cantidad de grupos a formar. Para ello se utiliza como referencia el gráfico de reducción de inercia sobre número de clúster, en el cual es ideal encontrar un tamaño en el que aumentar en una unidad la cantidad de clústers, no producirá un cambio tan alto en la reducción de la inercia.

```
hClustEvents <- hirarCluster(distAllMatrix)
```

Primero debemos indicar el número de grupos máximo, hasta donde se extenderá el gráfico de relevancia; este valor puede depender del número de datos que tengamos y un valor de referencia donde sospechemos que se encuentra el número correcto de clústers a formar. En nuestro ejemplo asignaremos 20, pero podemos cambiar este valor si no alcanzamos a definir una cantidad clara en el primer ensayo.

```
## Choose the maximun number of cluster to expand the graphic:
## 20
```



En nuestro ejemplo podemos ver que un buen número de cluster a formar corresponde con 5, ya que en adelante se producen muy pequeños cambios en la reducción de inercia.

Dado que en este paso hemos tomado una decisión, debemos indicar que no queremos actualizar el gráfico de barras, no obstante tenemos el chance de indicar un nuevo valor de clúster a expandir en el gráfico en caso de que optemos por la opción NO.

```
## Do you want update the barplot? Y/N
## N
```

Una vez se ha indicado que no se va a realizar una actualización del diagrama de barras, debemos señalar que queremos conformar un total de 5 clusters.

```
## Number of cluster:
## 5
```

Finalmente se acomodan los datos resultantes y se almacenan dentro de un archivo `RData`.

```
save(distAllMatrix,file = "distMatrixCluster.RData")

IdEvent <- names(evenF)

eventosClasificados <- data.frame(base_eventos,Cluster=hClustEvents)

write.csv(eventosClasificados,"eventsClasificated.csv")
```

En la carpeta donde está ubicado el espacio de trabajo pueden consultarse gráficos descriptivos por cluster los cuales se han generado dentro del directorio `AllCluster`.

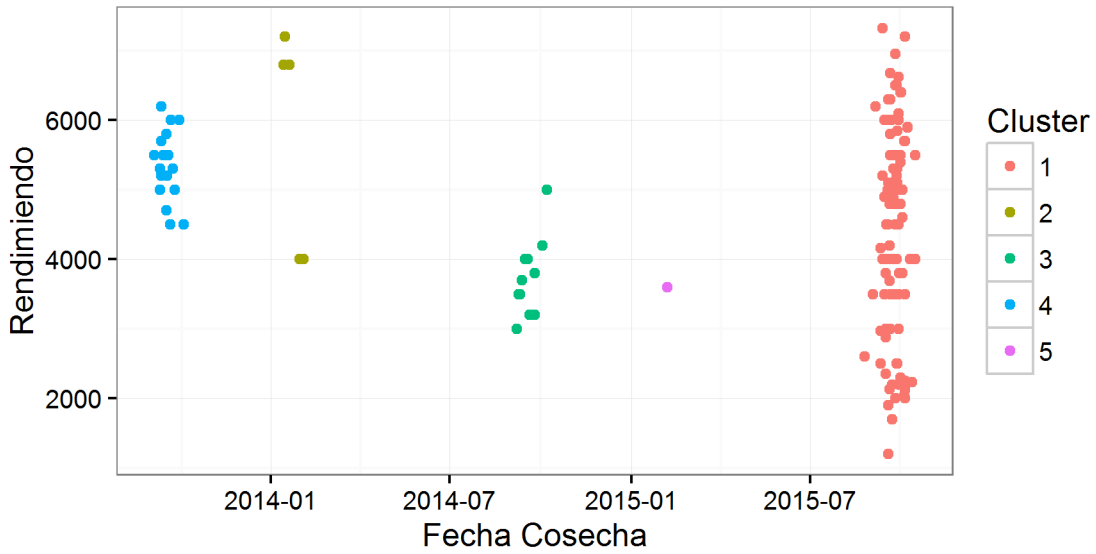
Gráficos exploratorios

En la carpeta donde está ubicado el espacio de trabajo pueden consultarse gráficos descriptivos por cluster los cuales se han generado dentro del directorio `AllCluster` una vez se ha ejecutado el comando `rsmnClustClima`.

```
rsmnClustClima(evenF,hClustEvents,varsI)
```

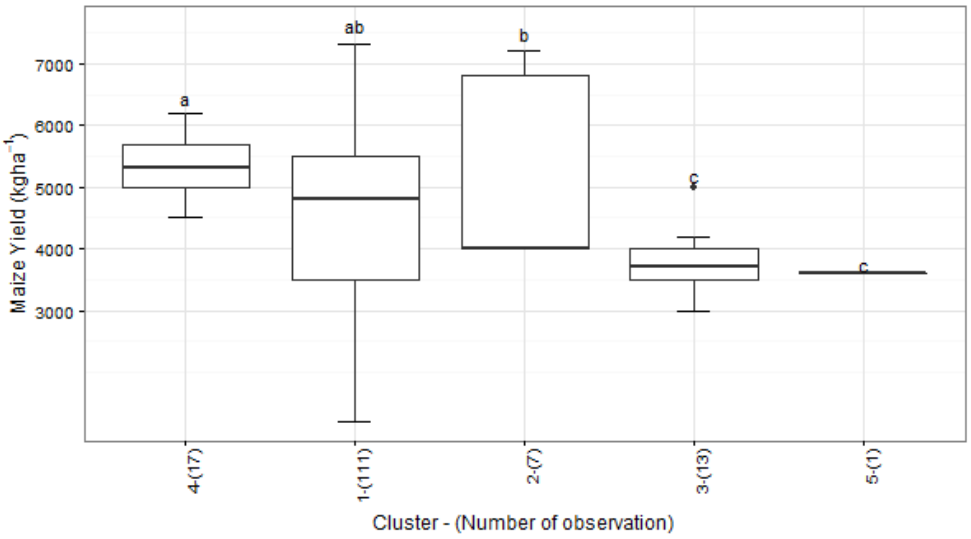
El siguiente gráfico permite observar la distribución de los cluster conformados respecto a las fechas de cosechas y rendimientos alcanzados.

```
m=ggplot(eventosClasificados,aes(FECHA_COSECHA,Rendimiento,
  colour=as.factor(Cluster)))+geom_point()+theme_bw()+xlab("Fecha Cosecha")+
    ylab("Rendimiendo")+guides(colour=guide_legend(title="Cluster"))

ggsave("RendimientoFechaCluster.png",m,width =6 ,height = 3)
```



Finalmente también podemos generar unos gráficos boxplots de rendimiento para comparar las condiciones climáticas que más favorecieron al cultivo; el encabezado lleva una prueba de kruskal para definir las diferencias significativas que se presentan entre grupos.

```
h <- krusk.boxplot(baseConComparacion=eventosClasificados,vary = "Rendimiento",
                varx = "Cluster",
                ylabs=expression(paste("Maize Yield (kg.",ha^-1,")")),maxVar=0)

png("imagenes/boxplot_numDatosRend.png",height = 350, width = 600)
print(h)
dev.off()
```



Referencias

http://www.rdatamining.com/examples/time-series-clustering-classification

Giorgino, Toni "Computing and visualizing dynamic time warping alignments in R: the dtw package." *Journal of statistical Software* 31.7 (2009): 1-24.