



✂ ✂ ✂ ✂ ✂

# Editor GPS

Miriam Lerma

Abril 2021

✂ ✂ ✂ ✂ ✂

# Intro

En esta presentación verán como:

- Cargar datos de GPS
- Recortar periodos de un solo individuo  
Usando paquete sula  
Convertir a formato dia y hora
- Recortar periodos de múltiples individuos  
Usando paquete sula  
Usando función  
Usando iteración
- Exportar datos

## Ustedes

- Tienes una base de datos con datos de GPS
- Tienen conocimientos básicos de R, si no te recomiendo [🔗empezar por aquí](#)

# Motivación

- A menudo los GPS deben ser activados antes de colocarse, y son apagados tiempo después de ser recolectados.
- Por eso es importante registrar la hora de inicio (cuando estaba en el animal) y cuando fue retirado.
- Con esta información podemos editar nuestros datos colectados de GPS para que correspondan solo al periodo en el que estuvo en el animal.

Pero...



¿Como los recortamos?



# 1. Importar datos a R

Para esta parte necesitas saber cargar tus datos, si necesitas recordar como hacerlo, te recomiendo [🔗empezar por aquí](#).

# 1.1. Cargar datos

Para cargar datos de prueba.  
Puedes instalar el paquete 'sula'

```
remotes::install_github("MiriamLL/sula")
```

Cargar datos de un solo individuo

```
library(sula)  
GPS_01<-GPS_01
```

Cargar datos de diez individuos

```
library(sula)  
GPS_raw<-GPS_raw
```



## 2. Recortar usando un solo individuo

Para esta parte tus datos deben ser transformados al formato POSIXct, si necesitas más información [ve aquí](#).

## 2.1. Usando paquete sula.

Supongamos que queremos recortar periodos del track del GPS01 que se colocó a las 18:10:00 del 02/11/2017 y se retiró a las 14:10:00 el 05/11/2017.

Esta función te permite recortar periodos dentro de tus datos.

```
GPS_recortado<-recortar_periodo(GPS_data=GPS_01,  
                                inicio='02/11/2017 18:10:00',  
                                final='05/11/2017 14:10:00',  
                                dia_col='DateGMT',  
                                hora_col='TimeGMT',  
                                formato="%d/%m/%Y %H:%M:%S")
```

## El track original contenía 1038 filas y el track editado contiene 986 filas

Te regresa un dataframe solo con los periodos de interés.

Listo!



## 2.2. Convertir a formato día y hora

Si quieres hacerlo lo mismo pero paso a paso.

Para convertir a formato día y hora, debes unir las columnas de día y hora y usar el argumento POSIXct para que R entienda que son día y hora.

Unir columnas

```
GPS_01$diahora<-paste(GPS_01$DateGMT, GPS_01$TimeGMT)
```

Transformar formato

```
GPS_01$diahora<-as.POSIXct(strptime(GPS_01$diahora,  
                                     "%d/%m/%Y %H:%M:%S"), "GMT")
```



## 2.2. Convertir a formato día y hora

Supongamos que queremos recortar periodos del track del GPS01 que se colocó a las 18:10:00 del 02/11/2017 y se retiró a las 14:10:00 el 05/11/2017.

Podemos escribir manualmente la hora de inicio y final.

```
Hora_inicio<- '02/11/2017 18:10:00'  
Hora_final<- '05/11/2017 14:10:00'
```

Ambas horas se debe convertir a formato día y hora para que R entienda que son fecha y tiempo.

```
Hora_inicio<-as.POSIXct(strptime(Hora_inicio, "%d/%m/%Y %H:%M:%S"), "GMT")  
Hora_final<- as.POSIXct(strptime(Hora_final, "%d/%m/%Y %H:%M:%S"), "GMT")
```

Aparecerán Hora\_inicio y Hora\_final en tu **environment** en la sección de **values**

## 2.3. Errores comunes

Presta mucho atención a como tienes ordenados tus datos.

- Verifica si usas / o - para separar tus fechas, o que otro separador puede existir en tus datos.

Para el formato '21-12-2017 05:30:01'.

```
format = "%d-%m-%Y %H:%M:%S"
```

Para el formato '2017/12/21 05:30:01'.

```
format = "%Y/%m/%d %H:%M:%S"
```

## 2.4. Eliminar periodos dentro de nuestros datos

Paso a paso, para eliminar periodos dentro de nuestros datos.  
Se puede utilizar la función `mutate` de la libreria `dplyr` dentro de `tidyverse`, para agregar una columna con Y o N, siendo N el periodo que queremos eliminar.

```
library(tidyverse)
```

No es necesario crear esa columna, pero es útil para verificar primero que funcione correctamente.

```
GPS_01<-mutate(GPS_01,  
               track = ifelse(GPS_01$diahora >= Hora_inicio &  
                             GPS_01$diahora <= Hora_final,  
                             "Y", "N"))
```

Ahora tendras en tu `GPS_01` una nueva columna llamada `track`.

## 2.4. Eliminar periodos dentro de nuestros datos

Usando información de la columna **track** se pueden filtrar para quedar solo con el periodo que interesa.

**Opción 1:** usando la función **filter**

```
GPS01_opcion1<-GPS_01 %>% filter(track=='Y')
```

**Opción 2:** usando la función **subset**

```
GPS01_opcion2<-subset(GPS_01,GPS_01$track=='Y')
```

Nota que el numero de observaciones se reduce y las dos opciones dan resultados idénticos.





# 3. Recortar multiples individuos

Como usar funciones para cortar periodos en nuestros datos de GPS cuando tenemos muchos individuos.

## 3.1. Datos de libreta

El paquete `sula` provee con datos de ejemplo de datos de libreta.  
Para usar las funciones te recomiendo tener los mismos nombres en las columnas.

```
library(sula)
Notas<-Notas
head(Notas)
```

```
##      IDs      Hora_inicio      Hora_final
## 1 GPS01 03/11/2017 08:00:00 05/11/2017 14:00:00
## 2 GPS02 03/11/2017 09:00:00 05/11/2017 15:00:00
## 3 GPS03 03/11/2017 10:00:00 05/11/2017 16:00:00
## 4 GPS04 03/11/2017 11:00:00 05/11/2017 17:00:00
## 5 GPS05 03/11/2017 12:00:00 05/11/2017 18:00:00
## 6 GPS06 03/11/2017 13:00:00 05/11/2017 19:00:00
```

## 3.2. Usando paquete sula

Puedes recortar periodos en los viajes.  
Para el ejemplo hay que tener dos data frames:

- **Datos de GPS** incluyendo las columnas 'DateGMT','TimeGMT' y 'IDs'.  
Si no tienen estos nombres favor de renombrarlas.
- **Datos de campo** debe incluir las columnas 'IDs', 'Hora\_inicio' y 'Hora\_final'.  
Si no tienen esos nombres favor de renombrarlas.

```
GPS_recortados<-recortar_por_ID(GPS_data=GPS_raw,  
                                Notas=Notas,  
                                formato="%d/%m/%Y %H:%M:%S")
```

```
range(GPS_recortados$dia_hora)
```

```
## [1] "2017-11-03 08:02:18 GMT" "2017-11-26 22:59:56 GMT"
```

Listo!



## 3.3. Basicos de una función

También se puede hacer lo mismo, pero paso a paso usando una función. Una función nos permite ahorrarnos el tiempo de escribir todos los pasos.

Para crear una función se usa el argumento **function** y corchetes.

```
Nombre_de_la_funcion<-function() {}
```

...pero no entrare en detalles.

Si quieres aprender más sobre funciones [🔗](#) ve aqui



## 3.3. Función

Si no quieres crear tu propia función, usa la siguiente función pero asegúrate de tener:

1. Un data frame con tus datos de GPS con cinco columnas siendo la 5ta IDs y una columna **diahora** (en POSIXct).
2. Un data frame con tus datos de campo con el nombre **Notas** que contenga tres columnas (IDs, Hora\_inicio y Hora\_final).

```
cortar_viaje<-function(GPS_id=GPS_id){  
  Nombre<-as.character(GPS_id[1,5]) #5ta columna es IDs  
  Horas<-subset(Notas,Notas$IDs==Nombre)  
  Hora_inicio<-Horas[1,2] #2nda columna es Hora_inicio  
  Hora_final<-Horas[1,3] #3era columna es Hora_final  
  Hora_inicio<- as.POSIXct(strptime(Hora_inicio, "%d/%m/%Y %H:%M:%S"), tz="UTC")  
  Hora_final<- as.POSIXct(strptime(Hora_final, "%d/%m/%Y %H:%M:%S"), tz="UTC")  
  GPS_recortado<-mutate(GPS_id,track=ifelse(GPS_id$diahora>=Hora_inicio  
                                           GPS_id$diahora<=Hora_final,"Y", "N"))  
  GPS_recortado<-GPS_recortado %>% filter(track=="Y")  
  assign(paste0(Nombre, "_recortado"),GPS_recortado,envir=.GlobalEnv)}
```

Cuando cargas tu función te aparece el **environment** en la parte de **Functions**

## 3.4. cortar\_viaje

**Nota** Recuerda cambiar tus datos a formato fecha y hora.

```
GPS_raw$diahora<-paste(GPS_raw$DateGMT, GPS_raw$TimeGMT)  
GPS_raw$diahora<-as.POSIXct(strptime(GPS_raw$diahora, "%d/%m/%Y %H:%M:%S"))
```

Y cargar la libreria

```
library(dplyr)
```

En la dispositiva anterior creamos una funcion que se llama **cortar\_viaje**. Para que haga algo hay que darle argumentos. Por ejemplo, el nombre del objeto GPS01.

```
cortar_viaje(GPS_01)
```

Al final la función agrega un nuevo objeto a tu environment con el nombre del GPS y "\_recortado". En este caso creara **GPS01\_recortado**

## 3.4. cortar\_viaje

Listo!

```
range(GPS01_recortado$diahora)
```

```
## [1] "2017-11-03 08:02:18 GMT" "2017-11-05 13:56:13 GMT"
```

También se puede usar:

```
cortar_viaje(subset(GPS_raw, GPS_raw$IDs== 'GPS02' ))
```

Las dos opciones funcionan.

```
range(GPS02_recortado$diahora)
```

```
## [1] "2017-11-03 09:02:57 GMT" "2017-11-05 14:57:40 GMT"
```





## 4. Iteraciones

Como usar iteraciones cortar periodos en nuestros datos de GPS cuando tenemos muchos individuos en una lista. Si quieres aprender más sobre iteraciones [ve aquí](#)

## 4.1. Listas

Para hacer una iteración con los datos, lo primero es recordar que los datos deben estar en formato día y hora

```
GPS_raw$diahora<-paste(GPS_raw$DateGMT,  
                       GPS_raw$TimeGMT)  
  
GPS_raw$diahora<-as.POSIXct(strptime(GPS_raw$diahora,  
                                     "%d/%m/%Y %H:%M:%S"), "GMT")
```

Posteriormente, hay que crear una lista usando como referencia el ID.

```
Lista_GPS<-split(GPS_raw, GPS_raw$IDs)
```

Ahora aparecerá en tu **environment** Lista\_GPS, y este objeto es una **Large list**.

Podemos acceder a elementos individuales de esta lista. Por ejemplo:

```
GPS_01<-Lista_GPS[[1]]
```

## 4.2. Aplicar

Pero para no hacer individuo por individuo, podemos hacer una **iteración** que se aplique a todos los elementos de la lista

Si quieres usar esta **iteración**, solo asegurate de

- Tener una lista con los datos de tus GPS que contenga una columna IDs y **diahora** en formato POSIXct
- Tener una objeto llamado **Notas** con IDs, Hora\_inicio y Hora\_final

```
for( i in seq_along(Lista_GPS)){  
  GPS_id<-Lista_GPS[[i]]  
  Nombre<-as.character(GPS_id[1,5]) #columna IDs  
  Horas<-subset(Notas,Notas$IDs==Nombre) #Notas  
  Hora_inicio<- as.POSIXct(strptime(Horas[1,2], "%d/%m/%Y %H:%M:%S"), tz=  
  Hora_final<- as.POSIXct(strptime(Horas[1,3], "%d/%m/%Y %H:%M:%S"), tz=  
  Lista_GPS[[i]]<-dplyr::mutate(GPS_id,track= ifelse(GPS_id$diahora>=Hora_inicio,  
                                                    GPS_id$diahora<=Hora_final,"Y","N"))  
}
```

## 4.3. Data frame

Puedes volver a pegar los elementos de tu lista en un data frame.

```
GPS_df<- do.call("rbind",Lista_GPS)
```

Y quedarte solo con los periodos que te interesan.

```
GPS_recortados<-GPS_df %>% filter(track=='Y')
```

Listo!





## 5. Exportar

Para esta parte necesitas saber exportar tus datos, si necesitas recordar como hacerlo, te recomiendo [🔗empezar por aquí](#).



## 5.1. Comparar

Si quieres comparar los cambios y ver si todo funcionó. Usa `table` para ver el numero de observaciones. Si tenemos menos observaciones que en el archivo original. Eso significa que logramos cortar periodos en los tracks de manera **exitosa**.

Datos originales

```
table(GPS_raw$IDs)
```

GPS01	GPS02	GPS03	GPS04	GPS05	GPS06	GPS07	GPS08	GPS09	GPS10
1038	1049	1246	1031	962	1231	1004	1015	931	933

Datos recortados

```
table(GPS_recortados$IDs)
```

GPS01	GPS02	GPS03	GPS04	GPS05	GPS06	GPS07	GPS08	GPS09	GPS10
783	787	784	775	698	771	436	436	436	437



## 5.2. Exportar

De acuerdo a lo que quieras hacer, puedes exportar los GPS de manera individual

```
GPS_01
```

O puedes exportar todos los GPS recortados

```
GPS_recortados
```

Seleccionar la carpeta.

```
library(here)
```

```
## here() starts at C:/Users/Lerma/Documents/4Cursos/1Bobos/ClaseTracking/Curs
```

```
DatosFolder<-here::here("01Datos")
```

Seleccionar archivo y el nombre de tu nuevo data frame.

```
write_csv(  
  GPS_recortados,  
  file=paste0(DatosFolder, '/GPS_recortados.csv'))
```


# 6. Contacto

Recapitulando:

- Cargar datos de GPS
- Recortar periodos de un solo individuo  
Usando paquete sula  
Convertir a formato día y hora
- Recortar periodos de múltiples individuos  
Usando paquete sula  
Usando función  
Usando iteración
- Exportar datos

Para dudas, comentarios y sugerencias:  
Escríbeme a [miriamjlerma@gmail.com](mailto:miriamjlerma@gmail.com)

Si quieres dar créditos puedes usar:

- Lerma M (2021) Package sula. Zenodo. <http://doi.org/10.5281/zenodo.4682898>
- Lerma M, Dehnhard N, Luna-Jorquera G, Voigt CC, Garthe S (2020) Breeding stage, not sex, affects foraging characteristics in masked boobies at Rapa Nui. Behavioral ecology and sociobiology 74: 149  OpenAccess  
Los datos de prueba vienen de esa publicación.