**Creación de mapas**

**web de Leaflet en R**



[R](https://es.wikipedia.org/wiki/R_(lenguaje_de_programaci%C3%B3n)) es un lenguaje de programación enfocado al tratamiento de datos con fines estadísticos, pero también dispone de paquetes para el tratamiento de datos espaciales. De hecho hay una [estrecha relación entre R y GIS](https://mappinggis.com/2019/02/r-y-gis-que-es-r-y-su-relacion-con-los-sig/). Hoy exploraremos la posibilidad de utilizar [Leaflet](https://leafletjs.com/) en R, que nos permitirá publicar los datos geográficos utilizados el visor del RStudio.

[RStudio IDE](https://rstudio.com/products/rstudio/download/) es un conjunto de herramientas integradas diseñadas para ayudarnos a ser más productivos con R y con Python. Incluye una consola, un editor que admite la ejecución directa de código y una variedad de herramientas sólidas para trazar, ver el historial, depurar y administrar su espacio de trabajo.

[Leaflet](https://leafletjs.com/) es un librería JavaScript de caracter open-source para el desarrollo de mapas móviles e interactivos. Este es el sistema que usa la aplicación de [geoparques](https://geoparques.icanh.gov.co) del ICANH para consignar y graficar la información de sitios e investigaciones arqueológicas.

# Primeros pasos con Leaflet y R

Instalar R

<https://r-coder.com/instalar-r/>

Instalar RStudio IDE: para la versión gratuita actual visitar: <https://rstudio.com/products/rstudio/download/>

para descargar versiones anteriores ir a:

<https://rstudio.com/products/rstudio/older-versions/>

Una vez instalados, podemos abir RStudio y seguir los siguientes pasos:

1. Antes de cualquier cosa, siempre es mejor configurar el espacio de trabajo de R antes de comenzar un proyecto. Por ejemplo, creemos un nuevo RProject para seguir usando este proyecto en el futuro. La forma más sencilla de hacer esto es hacer clic en el pequeño cuadro azul en la barra de atajos de RStudio. O Archivo> Nuevo proyecto ... en la barra de menú.
2. Para empezar deben saber que para que la aplicación funcione correctamente debemos siempre cargar los paquetes que usaremos en cada proyecto, en este caso les sugiero cargar los siguientes uno a uno escribiendo o copiando el texto de cada uno en el editor de texto de la esquina superior izquierda y dando clic en el botón “Run” en la barra superior del editor antes de seguir con los demás:

install.packages("leaflet")

install.packages("Rcpp")

library(sp)

library(sf)

library(fs)

library(magrittr)

1. Ahora para que se hagan una idea de lo que haremos queremos nuestro primer mapa el marcador utilizaremos datos de la universidad para visualizar su ubicación. En el mismo editor donde cargamos los paquetes escribiremos ahora el siguiente código le haremos de nuevo al botón “Run” una vez le escribamos todo:

leaflet() %>%

addTiles() %>%

addMarkers(lng=-76.5306, lat=3.3420, popup="Universidad Icesi")

*Si todo salió bien deberían estar viendo en la esquina inferior derecha un mapa con el marcador de la universidad. Igualmente me interesa te empiezan a ver el lenguaje en el que la primera línea del código llamar el paquete, la segunda pones a operar la función de “Tiles” Y La tercera línea la función de “Markers” O marcadores. Lo que sigue después entre paréntesis son las coordenadas de longitud y latitud, seguidas del popup con el nombre que aparecerá cuando demos click.*

1. Bueno hasta aquí tenemos tan sólo un ejemplo pero ahora debemos trabajar con nuestros datos que vamos a usar del mapa colectivo. Por ahora solo descargaremos una capa de las que han creado en sus grupos. Vayan al mapa colectivo<https://www.google.com/maps/d/edit?mid=182iiQCtJBu7hWbMfWYzUIcdU3Om6MJpn&usp=sharing> y en los tres puntos junto al titulo del *Mapa 8 Décadas de Antropología en Colombia* den click a “Exportar a KML o KMZ”. Aquí tendremos la opción de exportar todo el mapa o sólo una de las capas. Por ahora trabajaremos con una sola, seleccionan entonces desplegable la capa que desean trabajar y dar al botón descargar. Una vez descargado trasladen el archivo de la carpeta de descargas a la carpeta del proyecto que hemos creado arriba.
2. Ahora debemos cargar los datos a nuestro proyecto en la interface del RStudio. Para hacer esto elijan un nombre con el que reconocerán sus datos y escriban el siguiente código en el editor de texto de la esquina superior izquierda. Una vez escrito estas líneas de código debemos correr los datos para comprobar que el archivo cargas satisfactoriamente. Para ello haremos clic en el botón “Run” en la barra superior del editor. Aquí pueden aparecer algunos errores, pero generalmente se tratan de problemas encontrando el archivo en la ruta indicada.

nombreelegido1 <- st\_read("nombredelacapadescargada.kml")

nombreelegido1

*Si todo sale bien en la consola que es la ventana inferior izquierda verán aparecer los datos describiendo las características del archivo. También verán aparecer en la parte superior derecha, en la pestaña de Environment la capa cargada describiendo el número de datos y variables. Si hacen clic en este archivo verán aparecer una pestaña extra en el editor a mano izquierda en donde pueden comprobar la información de nombre descripción y geometría de los datos de la capa descargar.*

1. El siguiente paso es designar la información que mostraremos en el mapa una vez demos clic en los marcadores, como en el ejemplo del popup de la universidad. Para ello escriban el siguiente código y pongámoslo a correr:

titulo=as.data.frame(nombreelegido1$Name)

titulo

Después de esta función debemos también extraer las coordenadas para separarlas en columnas independientes que faciliten nuestra georeferenciación. Corramos el siguiente código teniendo cuidado de escribir bien el “nombreelegido1” que hemos designado para nuestra capa entre paréntesis y asignando un nuevo nombre al archivo que resultará en este caso lo estoy llamando “nombreelegido2”:

st\_geometry\_type

nombreelegido2=st\_coordinates(nombreelejido1)

1. En la consola abajo veremos aparecer varias cosas al igual en la ventana superior derecha en el espacio de Environment veremos aparecer un archivo nuevo. Si lo abren verán también Como nos muestra ahora las coordenadas cada una en la columna separada, pero también con una columna Z que no tiene datos. Para facilitar el trabajo borraremos esa columna con el siguiente código. De nuevo elegiremos otro nombre para llamar este archivo que resultará aquí lo designó como “nombreelegido3”.

nombreelegido3=st\_zm(nombreelegido2, drop = TRUE, what = 'ZM')

*Si a estas alturas ya pueden entender el código lo que le estamos pidiendo el sistema que haga que este nuevo archivo llamado nombreelegido3 sea igual a dejar caer o “drop”, dejar caer qué? “what” la columna Z. Verán de nuevo a parecer un archivo con su nuevo nombre pero ahora sólo con las dos columnas de coordenadas.*

1. El siguiente paso será primero convertir el archivo a data frame, para luego darle nombre a esas columnas, y especificar que su contenido es numérico. Para ello corran uno a uno el siguiente código:

nombreelegido3=as.data.frame(nombreelegido3)

names(nombreelegido3)=c("lon","lat")

as.numeric(nombreelegido3$lon)

as.numeric(nombreelegido3$lat)

*Veremos de nuevo aparecer en la consola abajo datos numéricos de las coordenadas, al igual que nuestra caja de coordenadas ahora con los nombres en las columnas.*

1. Ahora con los datos ya discriminados por columnas y nombrados como coordenadas podemos replicar el segundo paso para crear el mapa, pero esta vez con los datos del mapa colectivo. Copien el siguiente código tomando precaución de llamar bien cada uno de los archivos que usaremos. Para claridad los datos los extraeremos del archivo de coordenadas “nombreelegido3”:

leaflet() %>%

addTiles() %>%

addCircleMarkers(data = nombreelegido3, lng= nombreelegido3$lon, lat= nombreelegido3$lat)

*Este código nos debería reproducir ahora las ubicaciones y puntos del mapa que hemos hecho en Google. Hasta aquí se preguntarán para qué tanta vuelta si llegamos al mismo lugar? O podrían pensar que esta forma de trabajar es contraria a la manera intuitiva en la que hemos procedido hasta hora (y estarían en lo cierto), pero a la larga esta nueva forma suele ser más eficiente para datos masivos y proyectos más complejos. En los pasos siguintes es donde el programa empieza mostrar sus ventajas.*

1. Primero, nos permite importar información que podríamos tener en un archivo Excel que definiéramos según criterios específicos. En segundo lugar, hemos visto como el mapa de Google también tiene limitaciones en términos la capacidad de graficar la información de manera clara. Este es un asunto sumamente importante si queremos hacer que la información sea útil para otros y que logre su objetivo comunicativo o informativo. Para que vean a que me refiero copiamos el siguiente código que nos permitirá ahora agrupar los marcadores y mostrarlos de una manera diferente.

leaflet() %>%

addTiles() %>%

addCircleMarkers(data = nombreelegido3, lng= nombreelegido3$lon, lat= nombreelegido3$lat, label = titulo$'nombreelegido1$Name', clusterOptions = markerClusterOptions())

*Noten en el código anterior que ya no sólo estamos llamando los datos de coordenadas del archivo “nombreelegido3,” Sino que le estamos perdiendo el programa que lo combine con la información textual de la columna de título de nuestro primer archivo y además le estamos pidiendo que corra la función de clusters lo que permitirá agrupar la información por proximidad.*

1. Para importar Excel

Import/Read Data from Excel (both xls and xlsx formats) into R using RStudio (readxl package);

1. How can leaflet pop-up labels be formatted in R?

leaflet() %>%

addTiles() %>%

addCircleMarkers(data = prueba, lng=prueba$y, lat=prueba$x,

popup=paste("Autor:", prueba$Autor, "<br>",

"Titulo:", prueba$Titulo, "<br>",

"Fuente:", prueba$Fuente, "<br>",

"Categoría:", prueba$Categoría, "<br>",

"Consultar:", "<a href='",prueba$Enlace,"'>enlace</a>"),

clusterOptions = markerClusterOptions)

1. Publicación web sencilla desde R

El paquete R knitr es un motor de programación alfabetizado de propósito general, con API ligeras diseñadas para brindar a los usuarios un control total de la salida sin un trabajo pesado de codificación. Combina muchas características en un solo paquete con ligeros ajustes motivados por mi uso diario de Sweave. Consulte la página de inicio del paquete para obtener detalles y ejemplos. Consulte las Preguntas frecuentes para obtener una lista de las preguntas más frecuentes (incluido dónde hacer preguntas).

Puede instalar la versión estable en CRAN:

install.packages ('knitr')

En RStudio, cree un nuevo documento de R Markdown seleccionando Archivo | Nuevo | R Markdown.

Haga clic en el botón Tejer HTML en la barra de herramientas del documento para obtener una vista previa de su documento.

En la ventana de vista previa, haga clic en el botón Publicar.

|  |
| --- |
|  |

REFERENCIAS:

<https://www.earthdatascience.org/courses/earth-analytics/get-data-using-apis/leaflet-r/>

<https://mappinggis.com/2019/04/creacion-de-visores-de-mapas-web-de-leaflet-en-r/>

<https://www.listendata.com/2020/12/leaflet-in-R.html>

<https://github.com/rstudio/leaflet/blob/master/inst/examples/marker-clustering.R>

<https://youtu.be/_dBoXvhUqT8>

<https://youtu.be/dBk8gGX1MNk>

<https://youtu.be/N-gEts4_sPs>

<http://enrdados.net/post/leaflet-mapas-dinamicos/>

<https://mitchellgritts.com/posts/load-kml-and-kmz-files-into-r/https://andrewbtran.github.io/NICAR/2017/maps/leaflet-r.html#multiple_locations_from_a_csv>

<https://api.rpubs.com/insight/leaflet>

<https://rstudio.github.io/leaflet/markers.html>

<https://rpubs.com/mattdray/basic-leaflet-maps>

<https://youtu.be/6Uz5KBZ8V5E>