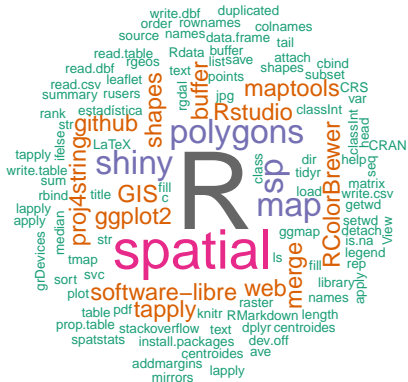


Taller de R aplicado a lo espacial



Richard Detomasi

rdetomasi@mides.gub.uy



Gabriela Mathieu

gmathieu@mides.gub.uy



Objetivos del taller

- ▶ Dar a conocer las potencialidades de R
- ▶ Presentar una introducción de R aplicado a lo espacial
- ▶ Ponernos en contacto con otras personas usuarias (actuales o futuras) de R

Secciones del taller

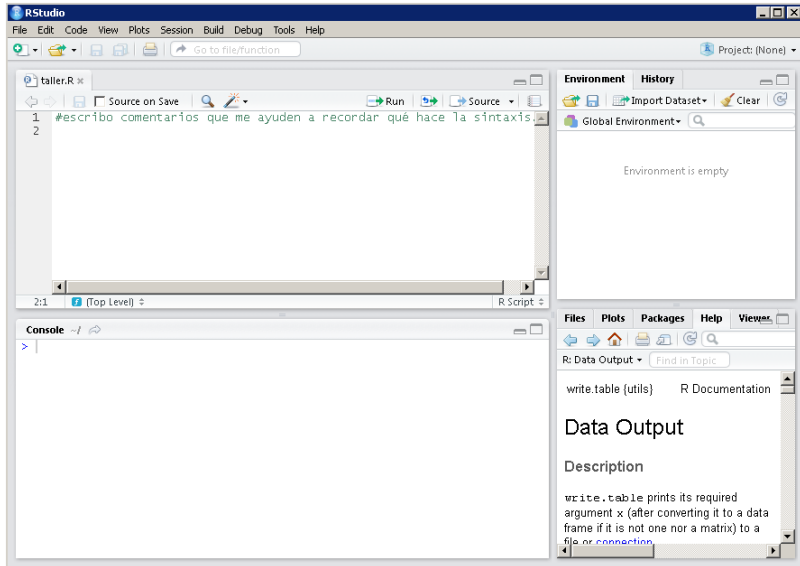
1. Presentación
2. Introducción a R
3. Presentación del ejercicio/problema
4. Objetos espaciales y métodos
5. Visualización de datos espaciales:
 - ▶ Mapas estáticos
 - ▶ Mapas web

Introducción a R

¿Qué es R?

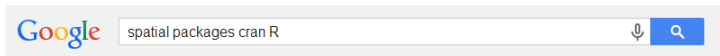
- ▶ Un lenguaje de programación y un programa estadístico.
- ▶ Es **software libre**: instalaciones ilimitadas, mejoras disponibles.
- ▶ Es **gratuito**, se descarga desde <http://www.r-project.org>.
- ▶ Hay una comunidad mundial que usa R y lo mejora constantemente ([paquetes](#)) y se ayudan entre sí: ([stackoverflow](#)), ([talkstats](#)), ([rusers](#)) y localmente ([meetup](#)).
- ▶ Orientado a objetos: datos, funciones, resultados, etc., se guardan en la memoria RAM en forma de objetos con un nombre específico, sin usar archivos temporales.

Un editor de código amigable para usar con R es **Rstudio**:



Introducción a R

- ▶ Las funciones tienen nombres descriptivos, se acompañan de ()
- ▶ **setwd**('home/username/Desktop/') # define el directorio de trabajo
- ▶ **getwd**() #indica el directorio de trabajo actual
- ▶ **help**(gBuffer) # brinda ayuda sobre el uso de la función, ídem **?gBuffer**
- ▶ **??**(gBuffer) # realiza una búsqueda de todas las funciones donde aparece esa(s) palabra(s)
- ▶ **apropos**('spatial') # lista funciones relevantes para el tema
- ▶ **help.search**('spatial') # similar al anterior
- ▶ **RSiteSearch**('spatial') # similar al anterior pero agrega una búsqueda desde el navegador de internet en el sitio web de R



Introducción a R: tips básicos
















- ▶ Los datos (cargados o generados) se guardan en objetos asignando un nombre con `=` o `<-`
casos `<- read.csv('home/Bases/datos.csv')`
- ▶ Al nombrar archivos u objetos, evitar usar tildes y ñ. No dejar espacio entre caracteres.
- ▶ Nombre de un objeto debe comenzar con una letra (A-Z, a-z)
- ▶ Puede incluir letras, dígitos (0-9), puntos (.) y guión bajo (_).
- ▶ R discrimina entre letras mayúsculas y minúsculas: `x` y `X` son objetos diferentes.
- ▶ Son nombres válidos **datos2016** o **datos_2016**, no así **2016datos** ni **datos-2016**

Introducción a R: tips básicos al trabajar con objetos

- ▶ Para ejecutar una línea de código, colocar el cursor sobre esa línea y ejecutar `Ctrl + ENTER`.
- ▶ Para ejecutar varias líneas de código, debemos seleccionarlas todas y ejecutar `Ctrl + ENTER`.
- ▶ El símbolo `>` en la consola, indica que R está listo para recibir un comando, mientras que si aparece el símbolo `+`, hay una sentencia no finalizada.
- ▶ Mensajes de error (**Error**): errores en la sintaxis (no ejecuta la/s líneas erróneas, pero sí el resto)
- ▶ Mensajes de precaución (**Warnings**): no necesariamente hay un error (ejecuta los comandos y solo te advierte de posibles inconvenientes).
- ▶ Importa saber qué tipo de objetos son los que estamos trabajando: no toda función se puede aplicar a cualquier tipo de objeto.

Introducción a R: Rstudio

- Atajos de teclado mientras el cursor se ubica en la consola (los dos últimos refieren a atajos desde el script):

Tecla	Descripción
	Vuelve a líneas de comando ejecutadas antes
	Avanza en el historial de comandos ejecutados
	Mueve el cursor hacia la izquierda
	Mueve el cursor hacia la derecha
	Mueve el cursor hacia el inicio de la línea
	Mueve el cursor hacia el final de la línea
	Sugiere funciones y comandos una vez escribamos el inicio de la palabra
 	Limpia la consola. Esto no elimina el historial.
 	Pasa el cursor desde la consola hacia la sintaxis
 	Pasa el cursor del script a la consola
 	Envía una línea o conjunto de comandos del script a la consola

Introducción a R: Organización de funciones

- ▶ Paquete: conjunto de funciones que se agrupan por realizar determinadas tareas afines.
- ▶ No es eficiente que todos los paquetes disponibles estén en la memoria activa de R: solo un número reducido se carga por defecto (ver **`save.session()`**, **`options()`**).
- ▶ Existen de tan variadas disciplinas que es muy probable que utilizemos relativamente pocos.
- ▶ Podemos consultar los paquetes disponibles escribiendo la función **`available.packages()`**. Actualmente son 9202 en el CRAN y otros tantos en ([github](#))!
- ▶ **`install.packages`** ('rgeos') # instala paquetes y dependencias (requiere comillas); se hace una única vez
- ▶ **`library`** (rgeos) # carga el paquete ya instalado, se hace cada inicio de sesión

Objetos: tipos de elementos y clases de objetos

Vector (contiene **elementos** de un **mismo tipo**)

[]

```
a=c(1,0,1,1,1,0) # vector numérico
```

```
b=c("1","0","1","1","1","0") # vector caracter
```

```
d=c(T,F,T,T,T,F) # vector lógico
```

```
f=factor(b,levels=c(1,0),labels=c('Sí','No')) # factor
```

Matriz (contiene **vectores** de una **misma clase**)

[,]

```
e=matrix(c(1,0,1,1,1,0),nrow=3) # matriz numérica
```

Data Frame (contiene **vectores** de **cualquier clase**)

\$

```
g=data.frame(e) # marco de datos, base de datos
```

Lista (contiene **objetos** de **cualquier clase**)

[[]]

```
e=list(g,f) # lista
```



Objetos espaciales: paquetes

- ▶ **sp**: clases y métodos para información espacial
- ▶ **rgdal**: interfaz a la biblioteca de procesamiento espacial gdal
- ▶ **rgeos**: procesamiento de vectoriales
- ▶ **maptools**: funciones de mapeo
- ▶ **tmap**: funciones de mapeo
- ▶ **ggplot2**: funciones gráficas
- ▶ **ggmap**: ploteo de mapas
- ▶ **leaflet**: biblioteca JavaScript para mapas interactivos
- ▶ **dplyr** y **tidyr**: manipulación de los datos
- ▶ **RColorBrewer**: define paletas de colores
- ▶ **classInt**: clasifica variables en intervalos

Objetos espaciales: clases

- ▶ SpatialPoints / SpatialPointsDataFrame / SpatialMultiPoints / SpatialMultiPointsDataFrame
- ▶ SpatialPixels / SpatialPixelsDataFrame / SpatialGridDataFrame / SpatialGridDataFrame
- ▶ Line / Lines / SpatialLines / SpatialLinesDataFrame
- ▶ Polygon / Polygons / SpatialPolygons / SpatialPolygonsDataFrame

Objetos espaciales: código

Paquete **sp** convierte objeto con coordenadas a objeto espacial

```
x_sp <- SpatialPoints(x[,1:2], proj4string = CRS('+init=epsg:32721'))
```

```
x_spdf <- SpatialPointsDataFrame(x_sp , CRS('+init=epsg:32721')), x,  
match.ID = T)
```

Paquete **maptools** carga y guarda un .shp

```
x_spdf <- readShapeSpatial('archivo shp', CRS('+init=epsg:32721'))
```

```
writePolyShape(objeto_estpacial, 'archivo shape')
```

Paquete **rgdal** carga y guarda un .shp

```
shape <- readOGR(dsn = 'ruta', layer = 'archivo shape', p4s =  
CRS('+init=epsg:32721'))
```

```
writeOGR(objeto _espacial, dsn = 'ruta', layer = 'nombre del shape', driver =  
'ESRI Shapefile')
```

Objetos espaciales: estructura

Slots básicos de un objeto espacial (sp):

- ▶ **@data** # la tabla de los objetos espaciales que tienen 'DataFrame'
- ▶ **@coords** # lista de coordenadas
- ▶ **@bbox** # límites de extensión
- ▶ **@proj4string** # Sistema de coordenadas

Para un `SpatialPolygonsDataFrame` en lugar del **@coords** hay un slot **@polygons** con 5 subslots:

- ▶ **labpt** # setea la posición para etiquetas (en este caso el centroide, vease función **coordinates()**)
- ▶ **area** # indica el área del polígono
- ▶ **hole** # definición lógica para presencia de huecos al interior del polígono
- ▶ **ringDir** # dirección en la que ordena los nodos de los huecos
- ▶ **coords** # y obviamente las coordenadas

Objetos espaciales: métodos

- ▶ **summary**(x) # presenta un resumen del objeto
- ▶ **bbox**(x) # brinda una matriz con las coordenadas de los límites de la capa (mín y máx de 'x' e 'y')
- ▶ **coordinates**(x) # brinda una matriz con todos los pares de coordenadas espaciales
- ▶ **geometry**(x) # recorta el data.frame, devolviendo solo las geometrías de los objetos
- ▶ **y = spTransform**(x, CRS('+proj=longlat +datum=WGS84'))
convierte o transforma de un sistema de coordenadas de referencia (CRS) a otro (requiere instalar paquete **rgdal**)

Ejercicio

- ▶ ¿Cuál es la cobertura de los SOCATs y sus Áreas Territoriales (AT)?
- ▶ ¿Que criterios de proximidad deben ser los que utilice esta política social?

Exploración de los datos

- ▶ Información georreferenciada
 - ▶ hogares potenciales usuarios de SOCATs: archivo Rdata
 - ▶ SOCATs: archivo csv
 - ▶ Áreas Territoriales: archivo shp
 - ▶ Municipios: archivo shp



Sintaxis 1

Objetos espaciales: métodos

- ▶ **over**(x, y) # devuelve un índice o atributos de los 'y' correspondientes (intersectados) con las locaciones espaciales de los 'x';
- ▶ **gBuffer**(x, byid ,id ,width) # genera un área de influencia
- ▶ **gUnaryUnion**(AT2,id) # disuelve geometrías por una variable
- ▶ **aggregate**(x = y['id'], by = polygon, FUN = length) # devuelve un resumen basado en variables por alguna variable de agrupamiento.
- ▶ **spsample**(x) # genera aleatoriamente coordenadas de puntos en el espacio continuo de SpatialPolygons, a gridded area, o a lo largo de SpatialLines;
- ▶ **splot**(x) # plotea atributos, permitiendo la combinación con otros tipos de objetos espaciales (points, lines, grids, polygons), y permite ploteo de múltiples atributos condicionales.

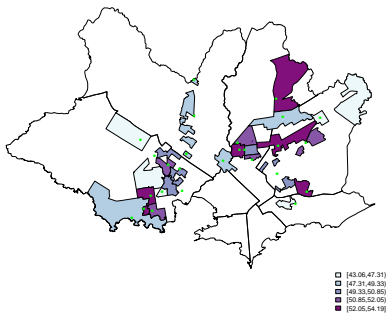


Objetos espaciales: Otros paquetes

- ▶ **spatstat**: permite análisis de patrones de puntos espaciales.
- ▶ **stplanr**: permite objetos de clase 'SpatialLinesNetwork' basado en objetos definidos en **sp** y **igraph**, y permite análisis de redes. Ver también: **shp2graph**.
- ▶ **cleangeo**: permite inspeccionar objetos espaciales, facilitando la evaluación y corrección de los errores topológicos y geometrías inválidas.
- ▶ **raster**: permite trabajar con grandes rasters, y extiende las opciones de herramientas de análisis.
- ▶ **spatial.tools** y **rasterVis**: complementos del paquete **raster**.
- ▶ **micromap**: genera micromapas combinados usando **ggplot2**.
- ▶ **recmap**: permite cartogramas rectangulares
- ▶ **spacetime**: extiende las clases definidas en **sp** para información espacio-temporal.
- ▶ **Grid2Polygons**: convierte un 'SpatialGridDataFrame' a 'SpatialPolygonsDataFrame'.

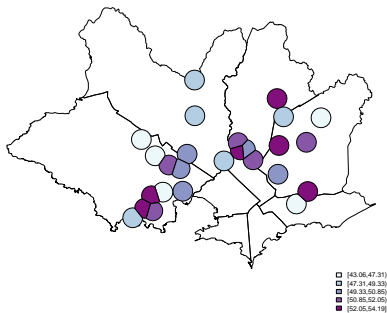
Mapas estáticos

Áreas atención SOCAT en Montevideo



Escenario 1

Áreas atención SOCAT en Montevideo



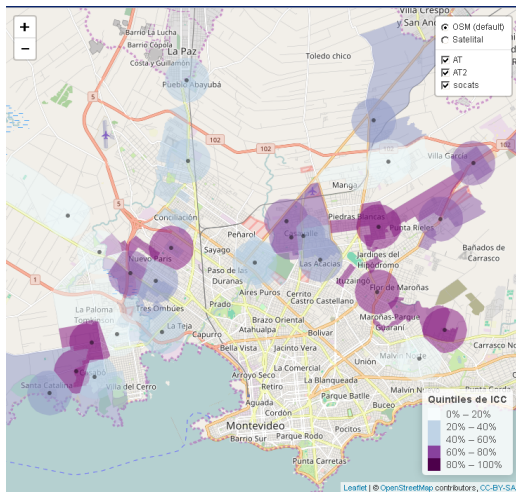
Escenario 2

Mapas web

Leaflet es una de los paquetes más populares **open-source JavaScript** para mapas interactivos que permite:

- ▶ Zoom y paneo interactivos
- ▶ Componer mapas usando arbitrariamente combinaciones de:
 - ▶ Mosaicos de mapas (OSM por defecto)
 - ▶ Marcadores
 - ▶ Polígonos
 - ▶ Líneas
 - ▶ Pop-ups
 - ▶ GeoJSON
- ▶ Crear mapas desde la consola de R
- ▶ Renderizar objetos espaciales del paquete **sp**, o **data frames** con columnas de latitud y longitud.
- ▶ Usar los límites del mapa e interacciones con el mouse, dentro de la lógica **Shiny**.

Mapas web



Shiny

Shiny es un paquete de R que permite realizar aplicaciones web interactivas (apps) sin necesidad de usar otro lenguaje de programación (aunque puede incluirlos).

- ▶ La aplicación tiene 2 componentes:
 - ▶ Código de la interfaz de usuario (iu.R): controla el diseño de la aplicación.
 - ▶ Código del servidor (server.R): contiene las instrucciones que el equipo necesita para construir su aplicación.
- ▶ La aplicación se ejecuta con la función `runApp()`
- ▶ Rstudio ofrece alojamiento para las apps
- ▶ Este es un [resumen](#) interesante sobre shiny



Ejemplo taller

Aplicación shiny

Este visualizador web es un ejemplo sencillo de cómo utilizar este paquete junto a leaflet.

Se incluye un panel a la izquierda usando un poco de código HTML

Esta app corre localmente a menos que se aloje en un servidor

[App con Shiny](#)

