Sistemas de información geográfica. TP1: Geoprocesos con datos vectoriales.

Marcos May

Indice

Objetivos 1
Tipo de Vectores. Carga y exploración de datos 2
Puntos 3
Líneas 7
Polígonos 7
Visualización interactiva con `mapview()` 10
Geoprocesos 12
Unión espacial con `st_join()` 13
Filtrado espacial `st_filter()` 18
Áreas de influencia (buffer) con `st_buffer()` 19
Intersección espacial entre capas: `st_intersection()` 20
Fusion de geometrías con `st_union()` 22
Excluir áreas superpuestas: `st_difference()` 22

Objetivos

Este trabajo se propone presentar cómo son tratados los **objetos espaciales vectoriales en R**, principalmente a través del uso de la librería `{sf}` dentro del ecosistema `{tidyverse}`. Se abordan conceptos fundamentales sobre el formato, la manipulación y la visualización de datos vectoriales, y se exploran algunas de las **operaciones básicas de geoprocesamiento.**y de visualización tanto dinámica (`mapview()`) como estática (`ggplot2 + geom_sf()`), aplicadas a datos reales provenientes de fuentes públicas.

Objetivo general

Manipular, visualizar y analizar datos geográficos vectoriales en R, utilizando herramientas del paquete `{sf}` y del ecosistema `{tidyverse}`.

Objetivos específicos

- Comprender el formato y la estructura de los objetos vectoriales en

  R.

- Abrir y trabajar con diferentes formatos de archivos geoespaciales (`.csv`, `.kml`, `.shp`, `.gpkg`).
- Explorar y visualizar datos espaciales mediante `mapview()` y `geom_sf()`.
- Aplicar filtros por atributos y realizar uniones espaciales.
- Revisar y utilizar fuentes de datos abiertas disponibles (Gobierno de la Provincia de Buenos Aires, INDEC, OpenStreetMap, RENABAP, entre otras).

Tipo de Vectores. Carga y exploración de datos

En los Sistemas de Información Geográfica (SIG), los datos vectoriales se utilizan para representar entidades geográficas mediante tres tipos básicos de geometrías:

- **Puntos:** representan ubicaciones precisas, como edificios, centros de salud o antenas.
- **Líneas:** representan elementos lineales, como caminos, ríos o redes de transporte.
- **Polígonos:** representan superficies delimitadas, como barrios, parcelas o áreas censales.

Cuando se cargan en R mediante la librería {sf}, estos datos vectoriales se transforman en data frames espaciales, también conocidos como geodataframes u objetos sf. Al igual que un data.frame o un tibble, estos objetos tienen filas (una por cada entidad geográfica) y columnas (atributos descriptivos). Lo que los distingue es la inclusión de una columna especial llamada geometry, que almacena la información espacial (coordenadas de puntos, líneas o polígonos).

Este diseño permite trabajar de forma integrada con los datos alfanuméricos y espaciales. Por ejemplo, al aplicar funciones como print() o head() se visualiza tanto la tabla como un resumen de la geometría. Además, los objetos sf son plenamente compatibles con el ecosistema {tidyverse}, lo que permite usar funciones como filter(), mutate(), group_by() o select() sobre capas espaciales. Estas operaciones pueden luego complementarse con funciones específicas

para análisis geográficos como st_join(), st_buffer(), st_intersection(), entre otras que se verán más adelante.

Una vez abiertos, los datos pueden explorarse visualmente de dos formas principales:

- **Visualización interactiva con `mapview()`:** permite inspeccionar los datos de forma dinámica, hacer zoom, ver atributos al pasar el cursor y superponer capas. Esta herramienta es ideal para la etapa exploratoria, aunque **no puede visualizarse directamente en archivos PDF**. Se recomienda su uso durante el trabajo en RStudio o en informes generados en formato HTML, donde la interactividad es compatible.
- **Visualización estática:**
  - **Con `plot()` (R base):** ofrece una forma rápida y sencilla de visualizar objetos `sf`. Es útil para hacer inspecciones básicas de los datos espaciales, como revisar geometrías, identificar errores topológicos o visualizar atributos de forma individual. No requiere paquetes adicionales.
  - **Con `ggplot2` y `geom_sf()`:** permite generar mapas reproducibles y de alta calidad, con mayor control sobre la estética, las escalas, los temas y la combinación con otros tipos de gráficos. Este enfoque es especialmente adecuado para informes formales, publicaciones científicas o presentaciones, y se desarrollará más adelante en el documento.

A continuación, se presentan ejemplos concretos de cada tipo de geometría (puntos, líneas y polígonos), utilizando fuentes de datos abiertas provistas por organismos oficiales (Gobierno de la Provincia de Buenos Aires, INDEC, RENABAP, OpenStreetMap, entre otros). Los datos provienen de distintos formatos geográficos (como .csv, .kml, .shp y .gpkg) y son abiertos en R mediante las funciones `st_read()` (cuando el archivo ya contiene geometría) o `st_as_sf()` (cuando la geometría se construye a partir de coordenadas). Se abordan algunas herramientas de exploración gráfica como `plot()` y `mapview()`.

Puntos

Capa: Establecimientos de Salud Públicos.

Fuente: Ministerio de Salud de la Provincia de Buenos Aires. (A través de Portal de Datos Abiertos de la Provincia de Buenos Aires).

Link: https://catalogo.datos.gba.gob.ar/dataset/establecimientos-salud/archivo/c52f9497-9eab-4ecd-a382-b4e4c6033a02

Formato: `.csv` (valores separados por comas)

El formato `.csv` (Comma-Separated Values) es un formato tabular ampliamente utilizado, aunque **no incluye información espacial directamente**. Para convertirlo en un objeto

espacial, es necesario indicar manualmente qué columnas contienen las coordenadas (en este caso: `long` y `lat`). Para ello se utiliza `st_as_sf()` de la librería `{sf}`.

```
salud<-
  read_delim(here("data/raw/salud/establecimientos_salud_publicos-2025.csv"),
    delim = ";", escape_double = FALSE, trim_ws = TRUE)
```

```
Rows: 3052 Columns: 20
-- Column specification --------------------------------------------------------
Delimiter: ";"
chr (13): cat, nor, nde, dom, nba, nrs, tel, mai, tes, Com, Dep, mod, Esp
dbl  (6): id, lat, long, cpd, cde, cp1
num  (1): cnr

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
salud_sf <-
  st_as_sf(salud,
           coords = c("long", "lat"),
           crs = 4326)
```

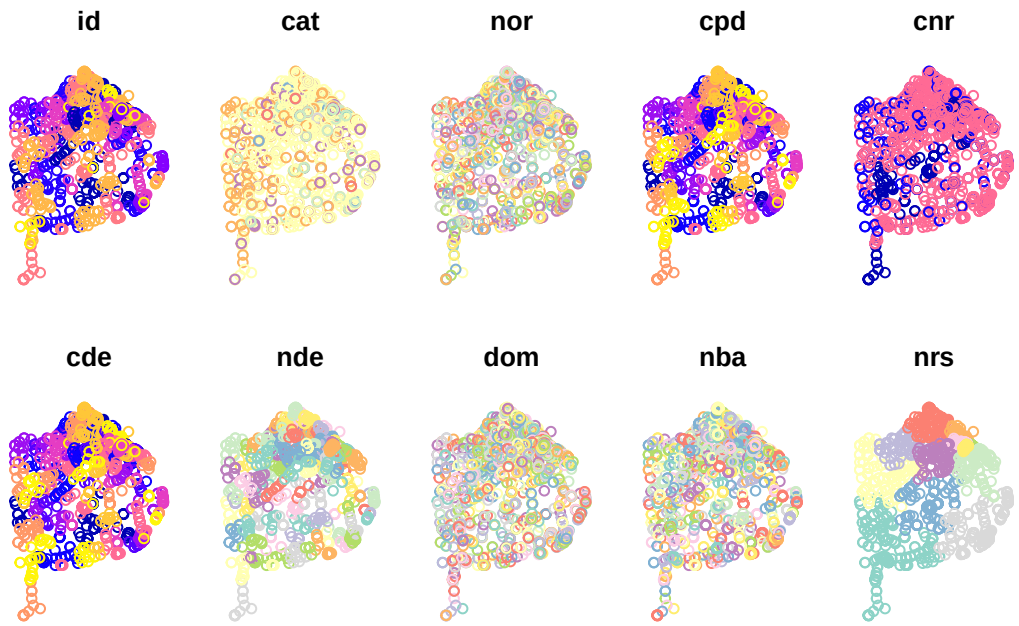
Visualización rápida con `plot()` (R base)

Una forma simple y directa de visualizar datos espaciales es utilizando la función `plot()` de R base, que es compatible con objetos `sf`. Esta función permite generar mapas estáticos sin necesidad de cargar paquetes adicionales de visualización.

Si usamos `plot()` directamente sobre el objeto `sf`, se generarán múltiples paneles, uno por cada variable del conjunto:

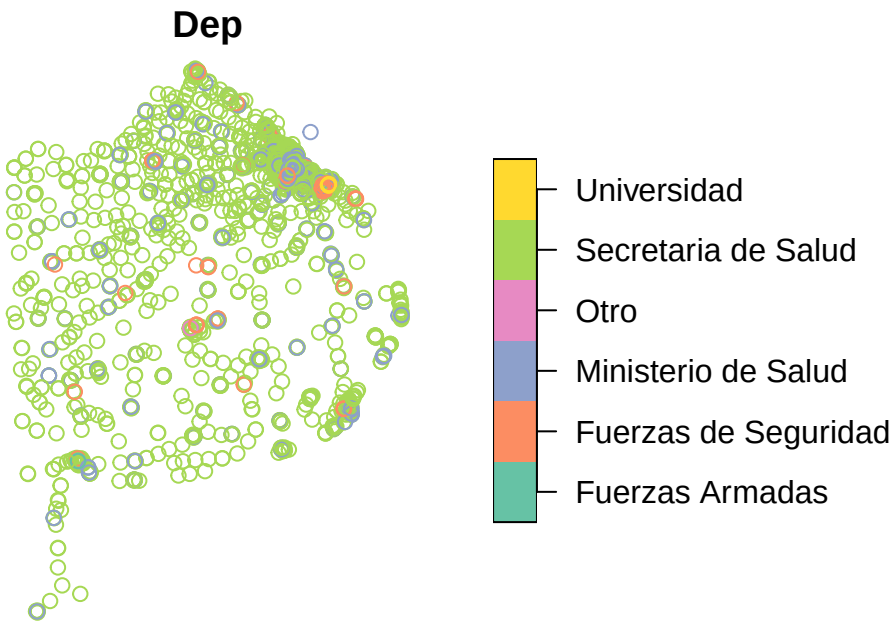
```
plot(salud_sf)
```

```
Warning: plotting the first 10 out of 18 attributes; use max.plot = 18 to plot
all
```



Podemos seleccionar una sola columna (atributo) utilizando el nombre de la variable. Por ejemplo, si queremos graficar únicamente la variable `Dep`, usamos:

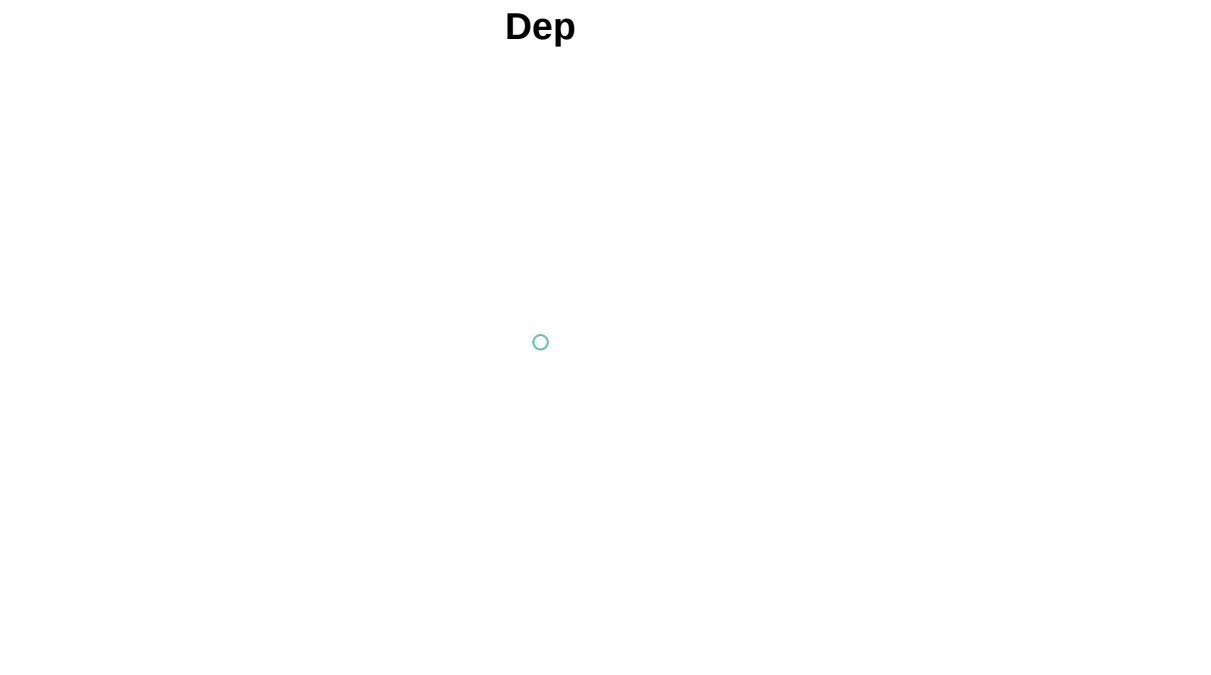
```
plot(salud_sf[,'Dep'])
```



Esto mostrará un mapa donde las geometrías se colorean según los valores de la variable `Dep`.

También podemos filtrar por fila, es decir, seleccionar una sola entidad geográfica (por ejemplo, un solo punto, línea o polígono) y graficarla:

```
plot(salud_sf[355,'Dep'])
```



En este caso se visualiza solo el punto ubicado en la posición 355 del dataframe. Este método cobra más relevancia cunado queremos explorar polígonos, por ejemplo para ver la forma de algún radio censal o barrio.

Capa: Establecimientos educativos.

Fuente: DGCyE. (A través de Portal de Datos Abiertos de la Provincia de Buenos Aires).

Link: https://catalogo.datos.gba.gob.ar/dataset/establecimientos-educativos

Formato: `.geojson` (GeoJSON)

El **formato `.geojson`** es una extensión del formato JSON especialmente diseñado para representar información geoespacial. Es un formato **ligero, basado en texto plano**, ampliamente utilizado para intercambio de datos en la web y compatible con una gran variedad de plataformas y librerías.

A diferencia del `.csv`, los archivos **`.geojson` ya contienen información geométrica estructurada**, y pueden abrirse directamente como objetos `sf` usando `st_read()`, sin necesidad de especificar columnas de coordenadas.

```r
escuelas <-
  st_read(here("data/raw/educacion/establecimientos-educativos-30062025.geojson"))
```

```
Reading layer `establecimientos-educativos-fecha' from data source
  `C:\Users\gigab\OneDrive\Documentos\GitHub\fahce_diplomatura_seminario_gis\data\raw\educaci
  using driver `GeoJSON'
Simple feature collection with 21568 features and 33 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: -63.38269 ymin: -40.8115 xmax: -56.67471 ymax: -33.29664
Geodetic CRS:  WGS 84
```

Líneas

Capa: Avenidas del partido de La Plata.

Fuente: OpenStreetMap (OSM), descargado mediante API.

Formato: .gpkg (GeoPackage).

El formato GeoPackage (.gpkg) es un estándar moderno y recomendado para trabajar con información geoespacial. A diferencia de archivos como .shp (que requieren múltiples archivos complementarios), un .gpkg almacena todos los datos en un único archivo, incluyendo geometría, atributos, proyecciones y múltiples capas.

Polígonos

Capa: Radios censales 2022

Fuente: Repositorio Institucional CONICET (Rodríguez, Gonzalo Martín)

Link: https://ri.conicet.gov.ar/handle/11336/238198

Formato: .shp (Shapefile).

El formato Shapefile (.shp) es uno de los formatos vectoriales más antiguos y extendidos en los Sistemas de Información Geográfica. Fue desarrollado por ESRI y permite almacenar información geoespacial de puntos, líneas o polígonos, junto con sus atributos.

Un shapefile no es un único archivo, sino un conjunto de al menos tres archivos obligatorios, que deben estar en la misma carpeta para que la información sea leída correctamente:

- `.shp`: contiene las geometrías (la representación espacial de los objetos).
- `.shx`: contiene un índice de las geometrías, que permite un acceso más rápido.

- `.dbf`: almacena los atributos (es una tabla en formato dBase, similar a una hoja de cálculo).

Además, pueden aparecer archivos adicionales:

- `.prj`: contiene la información de proyección (sistema de coordenadas).
- `.cpg`: indica la codificación de caracteres del archivo .dbf (útil para mostrar tildes y eñes).
- .qpj: una variante moderna del .prj, usada por algunos programas como QGIS.

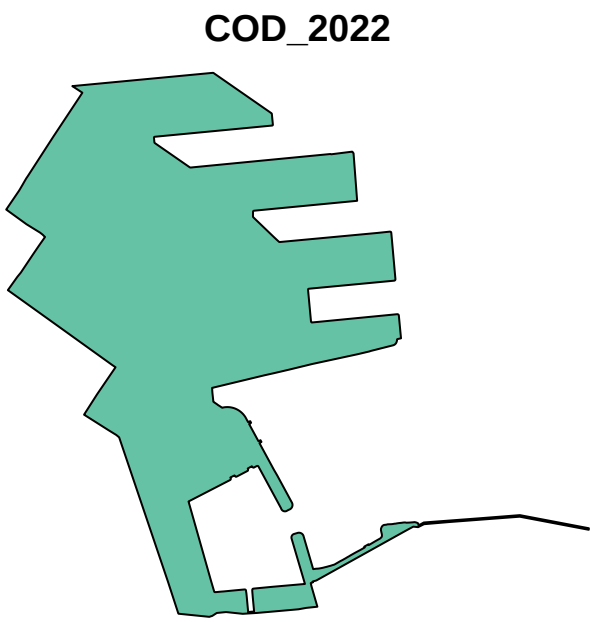
Para trabajar correctamente con un shapefile, es importante mantener juntos todos estos archivos. El archivo `.shp` por sí solo no es suficiente.

En este caso, se trata de la capa de radios censales publicada como insumo para el Censo Nacional 2022. Representa unidades geográficas en las que se organizan los operativos censales, y es una capa clave para análisis sociodemográficos.

```
radios <-
  st_read(here("data/raw/censo/radios_censales/Radios 2022 v2025-1.shp"))
```

```
Reading layer `Radios 2022 v2025-1' from data source
  `C:\Users\gigab\OneDrive\Documentos\GitHub\fahce_diplomatura_seminario_gis\data\raw\censo\r
  using driver `ESRI Shapefile'
Simple feature collection with 66502 features and 9 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -8237692 ymin: -30240970 xmax: -2922367 ymax: -2485225
Projected CRS: WGS 84 / Pseudo-Mercator
```

```
plot(radios[1,1])
```



Capa: Barrios populares (RENABAP)

Fuente: Registro Nacional de Barrios Populares (RENABAP), disponible en datos.gob.ar

Link: https://datos.gob.ar/dataset/habitat-registro-nacional-barrios-populares

Formato: `.gpkg` mal guardado (sin extensión)

Esta capa contiene los polígonos correspondientes a los **barrios relevados por el RENABAP**, que representan áreas urbanas informales reconocidas oficialmente por el Estado Nacional. Estos datos permiten identificar situaciones de vulnerabilidad habitacional y son clave en estudios urbanos, sociales y de accesibilidad.

El archivo provisto está en formato GeoPackage, pero fue **mal guardado sin extensión**, lo que genera una advertencia al intentar leerlo con `{sf}`. Aun así, `st_read()` logra interpretar correctamente el contenido si se especifica la ruta al archivo.

```
renabap <-
  st_read(here("data/raw/renabap/renabap-datos-barrios-gpkg"))
```

```
Warning in CPL_read_ogr(dsn, layer, query, as.character(options), quiet, : GDAL
Message 1: File
C:\Users\gigab\OneDrive\Documentos\GitHub\fahce_diplomatura_seminario_gis\data\raw\renabap\re
has GPKG application_id, but non conformant file extension
```

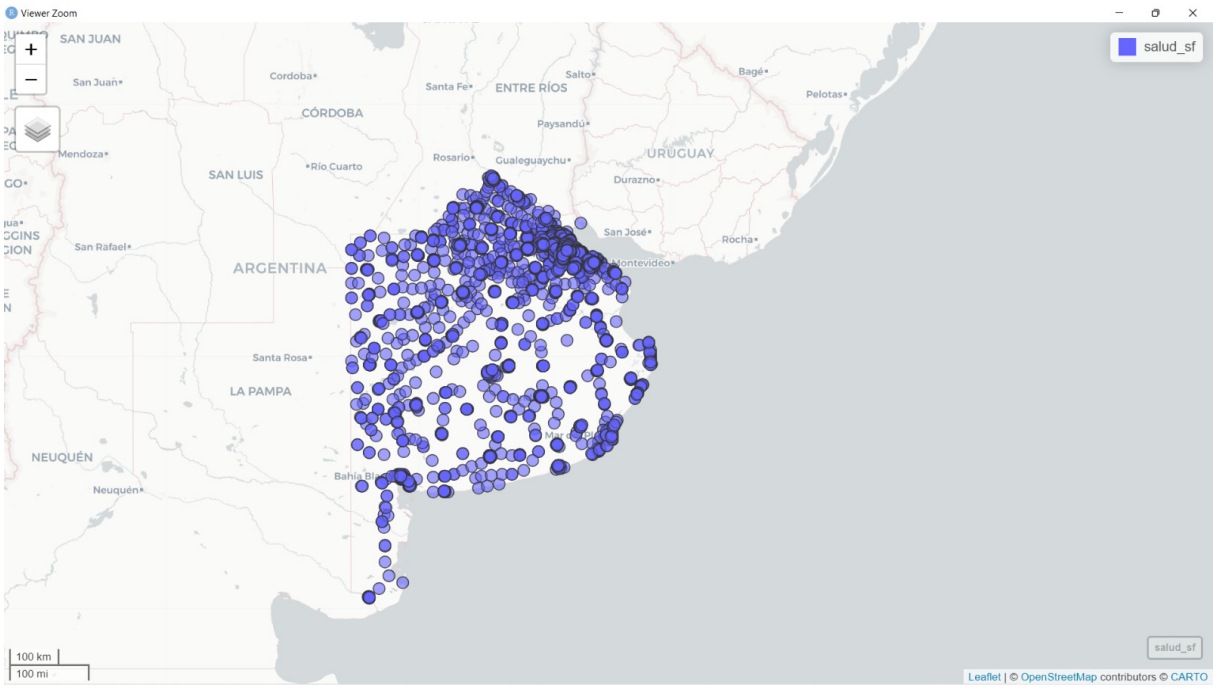
```
Reading layer `20231205_info_publica' from data source
  `C:\Users\gigab\OneDrive\Documentos\GitHub\fahce_diplomatura_seminario_gis\data\raw\renabap
  using driver `GPKG'
Simple feature collection with 6467 features and 17 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -72.21879 ymin: -54.82011 xmax: -53.64201 ymax: -22.03908
Geodetic CRS:  WGS 84
```

Visualización interactiva con `mapview()`

La función `mapview()` permite explorar datos espaciales de forma rápida e interactiva. A continuación se muestran algunos ejemplos con las capas cargadas:

1. Visualización básica (por defecto):

```
mapview(salud_sf)
```



2. Colorear por variable categórica (automático):

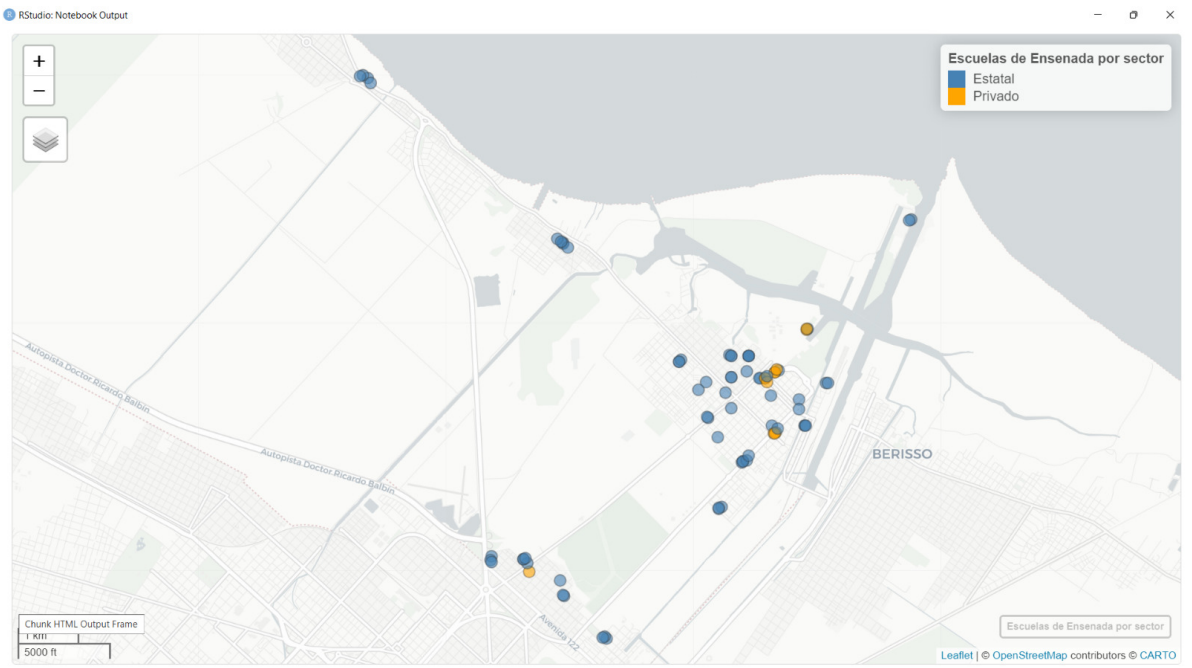
```
mapview(escuelas, zcol = "sector")
```

3. Aplicar filtro y personalizar colores por categoría:

```
cols_sector <- c("Público" = "steelblue", "Privado" = "orange")

escuelas |>
  filter(municipio_nombre == 'Ensenada' ) |>
  mapview(zcol = "sector",
          layer.name = "Escuelas de Ensenada por sector",
          col.regions = cols_sector,
          label = 'establecimiento_nombre',
          alpha = 0.5)

mapview(
  escuelas,
  zcol = "sector",
  col.regions = cols_sector,
  layer.name = "Escuelas por sector"
)
```



4. Superposición simple de capas:

```
mapview(renabap) + mapview(salud_sf)
```

5. Superposición con estilos personalizados:

```
mapview(
  renabap,
  col.regions = "tomato",
  color = "darkred",
  alpha.regions = 0.5,
  layer.name = "Barrios RENABAP"
) +
mapview(
  salud_sf,
  col.regions = "navy",
  layer.name = "Centros de Salud"
)
```

Se debe tener en cuenta que mapview() no se renderiza en documentos PDF, pero sí puede visualizarse en HTML o directamente desde RStudio. Además, al presionar el botón Export en la visualización interactiva, se puede exportar la visualización en HTML.

Geoprocesos

El **geoprocesamiento vectorial** consiste en aplicar operaciones espaciales sobre datos representados mediante geometrías como puntos, líneas o polígonos. Estas operaciones permiten transformar, combinar y analizar objetos espaciales para generar nueva información o extraer conclusiones útiles en contextos como salud pública, urbanismo, medio ambiente, transporte, entre otros.

El geoprocesamiento vectorial permite:

- Crear nuevas capas a partir de otras (por ejemplo, zonas de influencia).
- Analizar relaciones espaciales entre entidades (como qué puntos están dentro de qué polígonos).
- Extraer atributos espaciales (área, perímetro, centroides, distancias, etc.).
- Filtrar o recortar geometrías según criterios espaciales.
- Preparar datos para visualización o modelado.

Función	¿Qué hace?	Ejemplo básico
`st_filter()`	Filtra objetos de una capa en función de su relación espacial con otra capa. No agrega atributos, solo selecciona.	`st_filter(puntos, poligono, .predicate = st_within)`
`st_buffer()`	Genera un área de influencia o zona de proximidad alrededor de una geometría. Muy útil para analizar cobertura o impacto.	`st_buffer(salud_sf, dist = 500)`
`st_intersection()`	Devuelve la geometría que resulta de la superposición espacial entre dos capas.	`st_intersection(capa1, capa2)`
`st_union()`	Une múltiples geometrías en una sola, útil para agrupar o simplificar capas.	`st_union(capa)`
`st_difference()`	Calcula la diferencia entre dos geometrías, eliminando la parte común.	`st_difference(capa1, capa2)`
`st_centroid()`	Calcula el centro geométrico de cada objeto. Útil para representar polígonos como puntos.	`st_centroid(poligonos)`
`st_distance()`	Calcula la distancia mínima entre geometrías. Fundamental para análisis de cercanía.	`st_distance(puntos1, puntos2)`
`st_area()`	Calcula el área de polígonos. Requiere un CRS proyectado para obtener resultados en m^2.	`st_area(poligonos)`

Unión espacial con `st_join()`

La función `st_join()` permite realizar una **unión espacial de atributos** entre dos capas vectoriales. Es el equivalente espacial de un "left join" de `dplyr`, pero en lugar de emparejar filas por un valor común, las empareja por su **relación espacial** (como intersección, inclusión, contención, etc.). Esta herramienta es especialmente útil cuando queremos **agregar información de una capa a otra** en función de cómo se relacionan espacialmente.

Sintaxis básica

x: capa principal, que mantiene su geometría.

y: capa secundaria, de la que se traen atributos.

join: función que define la relación espacial (como st_intersects, st_within, etc.).

El resultado tendrá la geometría de x y los atributos agregados de y, según la relación espacial especificada.

Tipos de relaciones espaciales usadas en `st_join()`:

Relación espacial	Uso dentro de `st_join()`	¿Qué hace?
Intersección	`join = st_intersects`	Une atributos si las geometrías se superponen en alguna parte. Muy común para unir puntos dentro de polígonos.
Dentro de	`join = st_within`	Une solo si la geometría de `x` está completamente dentro de `y`.
Contiene	`join = st_contains`	Une si `x` contiene completamente a `y`.
Toca	`join = st_touches`	Une si las geometrías se tocan pero no se superponen.
Cruza	`join = st_crosses`	Une si las geometrías se cruzan pero no están contenidas entre sí.
Igual	`join = st_equals`	Une si las geometrías son idénticas.
Cercanía (distancia)	`join = st_is_within_distance`	Une si los objetos están dentro de una distancia específica. Debe usarse con el argumento adicional `dist = ...`.

Muchas de estas funciones de relación espacial (`st_intersects`, `st_within`, `st_contains`, etc.) pueden producir resultados similares o incluso iguales en algunos casos, especialmente cuando las geometrías son simples y no se superponen parcialmente.
Por lo general, se usa `st_intersects` como opción estándar porque captura cualquier tipo de superposición entre geometrías.
Las otras funciones se emplean en situaciones específicas donde se requiere definir con precisión cómo se relacionan los objetos espaciales (por ejemplo, cuando es importante que una geometría

esté completamente dentro de otra, o solo en contacto en el borde). Por eso, la elección de la función depende del objetivo del análisis y del nivel de detalle necesario.

> ⚠ Warning
>
> Antes de ejecutar un geoproceso entre dos capas es necesario que **ambas estén en el mismo sistema de referencia de coordenadas (CRS)**. Estas se pueden transformar utilizando la función `st_transform()`. Además, si al trabajar con **distancias, áreas, perímetros u otras medidas métricas** es fundamental que el CRS sea **proyectado** (por ejemplo, EPSG:5347), ya que los sistemas geográficos (como WGS84) no ofrecen unidades adecuadas para cálculos precisos.

Caso de estudio

Queremos identificar qué centros de salud (`salud_sf`) están ubicados dentro de los barrios populares (`renabap`). Para ello, primero es necesario asegurarse de que ambas capas estén en un sistema de coordenadas proyectado (en este caso, `EPSG:5347`), ya que las operaciones espaciales —como la intersección— requieren coherencia en el CRS.

Luego utilizamos `st_join()` con `st_intersects()` para realizar la unión espacial. El resultado lo guardamos en un nuevo objeto llamado salud_joined:

```
salud_joined <-
  st_join(salud_sf,
          renabap,
          join = st_intersects)
glimpse(salud_joined)
```

```
Rows: 3,052
Columns: 36
$ id                            <dbl> 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14~
$ cat                           <chr> "ESTABLECIMIENTO POLIVALENTE CON INTERN~
$ nor                           <chr> "HOSPITAL LOCAL MUNICIPAL GENERAL JOSE ~
$ cpd                           <dbl> 700014, 700022, 700057, 700065, 700073,~
$ cnr                           <dbl> 1.00601e+18, 5.00601e+18, 5.00601e+18, ~
$ cde                           <dbl> 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 14,~
$ nde                           <chr> "ADOLFO ALSINA", "ADOLFO ALSINA", "ADOL~
$ dom                           <chr> "AVELLANEDA 579   0", "9 DE JULIO  LOS ~
$ nba                           <chr> "CARHUE - A. ALSINA", "SAN MIGUEL ARCAN~
$ nrs                           <chr> "I", "I", "I", "I", "I", "I", "I", "I",~
$ cp1                           <dbl> 6430, 7403, 8185, 8185, 6341, 6430, 643~
$ tel                           <chr> "2936 432222 Int:", "292 4497051 Int:",~
$ mai                           <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, "ho~
```

```
$ tes                              <chr> "Municipal", "Municipal", "Municipal", ~
$ Com                              <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ Dep                              <chr> "Secretaria de Salud", "Secretaria de S~
$ mod                              <chr> "ASISTENCIAL CON INTERNACION", "ASISTEN~
$ Esp                              <chr> "GENERAL", "GENERAL", "GENERAL", "GENER~
$ geometry                         <POINT [m]> POINT (5255826 5882263), POINT (5~
$ id_renabap                       <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ nombre_barrio                    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ provincia                        <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ departamento                     <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ localidad                        <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ cantidad_viviendas_aproximadas   <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ cantidad_familias_aproximada     <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ decada_de_creacion               <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ anio_de_creacion                 <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ energia_electrica                <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ efluentes_cloacales              <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ agua_corriente                   <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ cocina                           <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ calefaccion                      <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ titulo_propiedad                 <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ clasificacion_barrio             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ superficie_m2                    <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
```

El objeto resultante, `salud_joined`, conserva la misma cantidad de filas que el objeto original `salud_sf` (esto no siempre es así con st_intersects, como veremos más adelante). A sus atributos se le suman las columnas provenientes de `renabap`, en aquellos casos donde existe una intersección espacial.

En las filas donde no hay intersección entre el centro de salud y algún barrio popular, las columnas correspondientes a renabap tendrán valores `NA`.

Probemos ahora a hacer la union espacial pero colocando como primer argumento (x) a la capa renabap:

```
renabap_joined <-
  st_join(renabap,
          salud_sf,
          join = st_intersects)
glimpse(renabap_joined)
```

```
Rows: 6,471
Columns: 36
```

```
$ id_renabap                      <int> 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13,~
$ nombre_barrio                   <chr> "Monterrey I", "Malvinas II", "Ferrovia~
$ provincia                       <chr> "Buenos Aires", "Buenos Aires", "Buenos~
$ departamento                    <chr> "Pilar", "La Plata", "La Plata", "La Pl~
$ localidad                       <chr> "Presidente Derqui", "José Melchor Rome~
$ cantidad_viviendas_aproximadas <int> 40, 290, 133, 122, 20, 12, 23, 200, 110~
$ cantidad_familias_aproximada   <int> 44, 319, 146, 134, 22, 13, 25, 220, 121~
$ decada_de_creacion              <chr> "Década 1990", "Década 1990", "Década 2~
$ anio_de_creacion                <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ energia_electrica               <chr> "Conexión regular a la red con medidor ~
$ efluentes_cloacales             <chr> "Desagüe a cámara séptica y pozo ciego"~
$ agua_corriente                  <chr> "Bomba de agua de pozo domiciliaria", "~
$ cocina                          <chr> "Gas en garrafa", "Gas en garrafa", "Ga~
$ calefaccion                     <chr> "Sin Datos", "Leña o carbón", "Leña o c~
$ titulo_propiedad                <chr> "NO", "NO", "NO", "NO", "NO", "NO", "NO~
$ clasificacion_barrio            <chr> "Asentamiento", "Asentamiento", "Asenta~
$ superficie_m2                   <int> 11674, 98093, 75887, 36889, 25768, 3424~
$ id                              <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ cat                             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ nor                             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ cpd                             <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ cnr                             <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ cde                             <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ nde                             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ dom                             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ nba                             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ nrs                             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ cp1                             <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ tel                             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ mai                             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ tes                             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ Com                             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ Dep                             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ mod                             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ Esp                             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ geom                            <MULTIPOLYGON [m]> MULTIPOLYGON (((5607158 61~
```

Observese que mientras el objeto original renabap antes del join tiene 6467 filas, luego de hacer el join pasa a tener 6471 filas. ¿De donde surgen estas 4 filas de diferencia?

La respuesta es que existen barrios populares con más de un centro de salud en su interior. En otras palabras, cuando un mismo barrio intersecta con más de un centro de salud, el resultado

del st_join() devuelve una fila por cada combinación de barrio y centro de salud intersectante, generando así múltiples registros para un mismo barrio.

```
renabap_joined |>
  st_drop_geometry() |>
  count(id_renabap, nombre_barrio, name = 'cantidad_centros_salud') |>
  filter(cantidad_centros_salud > 1)
```

```
  id_renabap   nombre_barrio cantidad_centros_salud
1        273  Santa Catalina                     2
2       1277 Tres de Febrero                     2
3       1479       San Alejo                     2
4       1577           Itatí                     2
```

Además de `st_join()`, existen varias funciones en `sf` que resultan fundamentales para el análisis espacial y la manipulación de datos vectoriales. Estas herramientas permiten trabajar con las geometrías y atributos de manera eficiente para responder a diferentes preguntas geográficas.

Filtrado espacial `st_filter()`

Cuando deseamos identificar qué elementos de una capa espacial se encuentran dentro de otra capa —sin tener en cuenta los atributos de ambas— podemos utilizar la función st_filter(). Esta función permite aplicar filtros espaciales basados en relaciones geométricas, como la intersección, contención o proximidad. Por defecto, st_filter() usa la relación espacial st_intersects(), lo que significa que devuelve aquellos elementos cuya geometría se intersecta con al menos una geometría de la capa de referencia.

```
st_filter(salud_sf,
          renabap) %>%
  glimpse
```

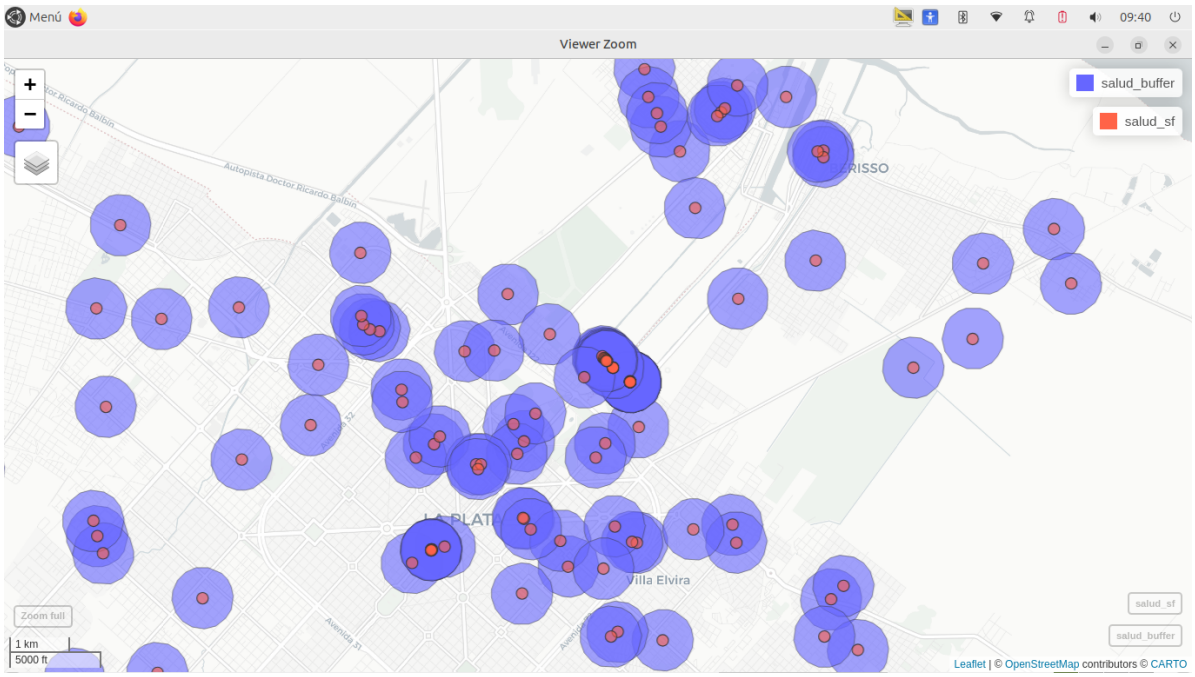
```
Rows: 115
Columns: 19
$ id       <dbl> 2010, 2216, 2998, 3000, 1318, 1328, 2265, 2701, 2726, 2728, 2~
$ cat      <chr> "C.A.P.S.", "C.A.P.S.", "C.A.P.S.", "C.A.P.S.", "C.A.P.S.", "~
$ nor      <chr> "CENTRO INTEGRADOR COMUNITARIO", "CENTRO PERIFERICO VILLA DEL~
$ cpd      <dbl> 55300127, 60200096, 87500181, 87500202, 41300284, 41300781, 6~
$ cnr      <dbl> 5.00655e+18, 5.00660e+17, 5.00688e+18, 5.00688e+18, 5.00641e+~
$ cde      <dbl> 553, 602, 875, 875, 413, 413, 623, 763, 770, 770, 770, 770, 4~
```

```
$ nde      <chr> "MONTE HERMOSO", "PATAGONES", "VILLARINO", "VILLARINO", "JUNI~
$ dom      <chr> "RIO SALADO  SARA CASEY 0", "7 DE MARZO   780", "MARTÍN FIERR~
$ nba      <chr> "MONTE HERMOSO", "CARMEN DE PATAGONES", "MEDANOS", "MAYOR BUR~
$ nrs      <chr> "I", "I", "I", "I", "III", "III", "IV", "IV", "IV", "IV", "IV~
$ cp1      <dbl> 8153, 8504, 8132, 8146, 6000, 6000, 2700, 2903, 2930, 2930, 2~
$ tel      <chr> NA, NA, "2927 433138 Int:", "291 4917281 Int:", "2362 1569048~
$ mai      <chr> NA, NA, "caps3med@gmail.com", "capsburatovich@yahoo.com.ar", ~
$ tes      <chr> "Municipal", "Municipal", "Municipal", "Municipal", "Municipa~
$ Com      <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
$ Dep      <chr> "Secretaria de Salud", "Secretaria de Salud", "Secretaria de ~
$ mod      <chr> "ASISTENCIAL SIN INTERNACION", "ASISTENCIAL SIN INTERNACION",~
$ Esp      <chr> "GENERAL", "GENERAL", "GENERAL", "GENERAL", "GENERAL", "GENER~
$ geometry <POINT [m]> POINT (5386290 5685017), POINT (5247071 5479907), POINT~
```

Áreas de influencia (buffer) con `st_buffer()`

`st_buffer()`genera áreas de influencia alrededor de geometrías espaciales. La distancia del buffer se define en las unidades del sistema de referencia de coordenadas (CRS). Es especialmente útil para análisis de proximidad.

```
salud_buffer <-
  st_buffer(salud_sf, dist = 500)
```



Intersección espacial entre capas: `st_intersection()`

La función `st_intersection()` permite calcular la intersección entre dos objetos espaciales, es decir, obtener únicamente las partes donde sus geometrías se superponen. El resultado es un nuevo objeto `sf` que contiene esas porciones compartidas con geometrías modificadas y atributos combinados de ambas capas.

Por ejemplo, si se intersectan los polígonos de una capa de barrios con buffers de 500 metros construidos alrededor de centros de salud, el objeto resultante incluirá únicamente los fragmentos de barrios que caen dentro de esas áreas de influencia. No se conserva la geometría completa de los barrios ni de los buffers, sino solo las partes en las que coinciden espacialmente.

Además de modificar la geometría, `st_intersection()` conserva los atributos de ambas capas originales. Es decir, cada fila del objeto resultante incluirá información tanto del barrio como del buffer (y por lo tanto, del centro de salud que lo originó). Esto lo diferencia de funciones como `st_filter()`, que filtran pero no modifican geometrías ni combinan atributos.

```
salud_intersection <-
  st_intersection(renabap,
                  salud_buffer)
```

```
Warning: attribute variables are assumed to be spatially constant throughout
all geometries
```

¿Qué ocurre con los nombres e índices al usar `st_intersection()`?

Cuando se aplica `st_intersection()`, el objeto resultante suele tener un número de filas mayor al de cualquiera de las capas originales. Esto se debe a que cada intersección individual entre geometrías se convierte en una nueva fila. Por ejemplo, si un buffer intersecta con tres barrios, eso generará tres filas diferentes en el resultado, una por cada combinación.

En este nuevo objeto:

- El índice de filas se reinicia y **ya no coincide** con los índices de las capas originales.
- Se **combinan los atributos de ambas capas**. Si ambas tienen columnas con el mismo nombre, `sf` renombra automáticamente una de ellas (por ejemplo, agregando `.1` o `.2`).
- En algunos casos, se agregan columnas especiales (como `row.id`, `origins`, `L1`, `L2`) para identificar de dónde proviene cada geometría.
- Las geometrías del resultado representan **solo la parte compartida** entre ambas capas.

Por estas razones, es importante inspeccionar el objeto resultante con funciones como `glimpse()`, `names()`, o `head()` para entender su estructura antes de usarlo en análisis posteriores.

```
salud_intersection %>% glimpse()
```

```
Rows: 1,746
Columns: 36
$ id_renabap                      <int> 652, 1233, 4917, 707, 709, 709, 2903, 4~
$ nombre_barrio                   <chr> "Costa Blanca", "Puertas del Sur", "Chi~
$ provincia                       <chr> "Buenos Aires", "Buenos Aires", "Buenos~
$ departamento                    <chr> "Bahía Blanca", "Bahía Blanca", "Bahía ~
$ localidad                       <chr> "Bahía Blanca", "Bahía Blanca", "Ingeni~
$ cantidad_viviendas_aproximadas  <int> 130, 600, 20, 45, 271, 271, 150, 205, 2~
$ cantidad_familias_aproximada    <int> 143, 660, 22, 50, 298, 298, 165, 226, 2~
$ decada_de_creacion              <chr> "Década 2010", "Década 2000", "Década 1~
$ anio_de_creacion                <int> 2012, NA, NA, NA, NA, NA, NA, 2015, NA,~
$ energia_electrica               <chr> "Conexión irregular a la red", "Conexió~
$ efluentes_cloacales             <chr> "Desagüe sólo a pozo negro/ciego u hoyo~
$ agua_corriente                  <chr> "Conexión irregular a la red de agua", ~
$ cocina                          <chr> "Gas en garrafa", "Gas en garrafa", "Ga~
$ calefaccion                     <chr> "Leña o carbón", "Sin Datos", "Leña o c~
$ titulo_propiedad                <chr> "NO", "NO", "NO", "NO", "NO", "NO", "NO~
$ clasificacion_barrio            <chr> "Asentamiento", "Asentamiento", "Asenta~
$ superficie_m2                   <int> 99605, 426846, 16073, 10340, 131937, 13~
$ id                              <dbl> 189, 189, 191, 196, 196, 200, 216, 219,~
$ cat                             <chr> "C.A.P.S.", "C.A.P.S.", "C.A.P.S.", "C.~
$ nor                             <chr> "SALA MEDICA CORONEL MALDONADO", "SALA ~
$ cpd                             <dbl> 5600221, 5600221, 5600263, 5600310, 560~
$ cnr                             <dbl> 5.00606e+18, 5.00606e+18, 5.00606e+18, ~
$ cde                             <dbl> 56, 56, 56, 56, 56, 56, 56, 56, 56, 56,~
$ nde                             <chr> "BAHIA BLANCA", "BAHIA BLANCA", "BAHIA ~
$ dom                             <chr> "GENERAL PABLO RICHIERI   2535", "GENER~
$ nba                             <chr> "BAHIA BLANCA", "BAHIA BLANCA", "INGENI~
$ nrs                             <chr> "I", "I", "I", "I", "I", "I", "I", "I",~
$ cp1                             <dbl> 8000, 8000, 8103, 8000, 8000, 8000, 800~
$ tel                             <chr> NA, NA, "291 4570474 Int:", "291 482162~
$ mai                             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ tes                             <chr> "Municipal", "Municipal", "Municipal", ~
$ Com                             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ Dep                             <chr> "Secretaria de Salud", "Secretaria de S~
$ mod                             <chr> "ASISTENCIAL SIN INTERNACION", "ASISTEN~
$ Esp                             <chr> "GENERAL", "GENERAL", "GENERAL", "GENER~
$ geom                            <GEOMETRY [m]> POLYGON ((5298017 5712185, ...~
```

```r
  salud_intersection %>%
  #st_drop_geometry() |>
  group_by(id_renabap) %>%
  mutate(  cantidad = n() ) %>%
  ungroup() %>%
  arrange(id_renabap) %>%
  filter(id_renabap == 57)
```

```
Simple feature collection with 4 features and 36 fields
Geometry type: GEOMETRY
Dimension:     XY
Bounding box:  xmin: 5693003 ymin: 6132203 xmax: 5693523 ymax: 6132580
Projected CRS: POSGAR 2007 / Argentina 5
# A tibble: 4 x 37
  id_renabap nombre_barrio provincia    departamento localidad
*      <int> <chr>         <chr>        <chr>        <chr>
1         57 Villa Alba    Buenos Aires La Plata     Villa Elvira
2         57 Villa Alba    Buenos Aires La Plata     Villa Elvira
3         57 Villa Alba    Buenos Aires La Plata     Villa Elvira
4         57 Villa Alba    Buenos Aires La Plata     Villa Elvira
# i 32 more variables: cantidad_viviendas_aproximadas <int>,
#   cantidad_familias_aproximada <int>, decada_de_creacion <chr>,
#   anio_de_creacion <int>, energia_electrica <chr>, efluentes_cloacales <chr>,
#   agua_corriente <chr>, cocina <chr>, calefaccion <chr>,
#   titulo_propiedad <chr>, clasificacion_barrio <chr>, superficie_m2 <int>,
#   id <dbl>, cat <chr>, nor <chr>, cpd <dbl>, cnr <dbl>, cde <dbl>, nde <chr>,
#   dom <chr>, nba <chr>, nrs <chr>, cp1 <dbl>, tel <chr>, mai <chr>, ...
```

Fusion de geometrías con `st_union()`

La función `st_union()` permite **fusionar geometrías** en una sola, disolviendo los límites entre elementos. Se utiliza, por ejemplo, para unir todos los buffers en una única área de cobertura.
No conserva atributos individuales, pero puede combinarse con `group_by()` y `summarise()` si se desea agrupar por una variable antes de la unión.

Excluir áreas superpuestas: `st_difference()`

La función `st_difference()` devuelve las partes de una geometría que **no se superponen** con otra. Es útil para identificar áreas **excluidas** por una capa, como las zonas de un barrio

que **no** están cubiertas por un buffer de salud.

> ⚠ Advertencia al usar st_difference()
>
> `st_difference(x, y)` calcula la parte de `x` que **no se superpone con** `y`. Si `y` tiene **múltiples geometrías**, la diferencia se aplica con **cada geometría por separado**, lo cual puede:
>
> - Generar **resultados inesperados**, como duplicaciones o geometrías partidas.
> - Aumentar la cantidad de filas del resultado.
> - Complicar la interpretación espacial.
>
> Es recomendable **fusionar previamente** las geometrías de `y` con `st_union()`, para que la diferencia se calcule respecto de una única geometría combinada.

```
renabap_lp <-
  renabap %>%
  filter(departamento == 'La Plata')

salud_lp <-
  salud_buffer %>%
  filter(nde == 'LA PLATA') %>%
  st_union() %>%
  st_as_sf()

resultado <- st_difference(renabap_lp, salud_lp)
```

```
Warning: attribute variables are assumed to be spatially constant throughout
all geometries
```

A diferencia de `st_intersection()`, no conserva atributos de la segunda capa, y las geometrías resultantes pertenecen solo a la primera.

La función `st_centroid()` calcula el **centro geométrico** de cada objeto espacial. Es útil para representar áreas con un punto, por ejemplo, al ubicar etiquetas o simplificar visualizaciones.

La función `st_distance()` calcula la **distancia** entre geometrías. Puede aplicarse entre puntos, líneas o polígonos, y devuelve una matriz de distancias por defecto. Si se desea conocer la distancia mínima entre dos capas, se puede usar con índices.

La función `st_area()` calcula el **área** de geometrías tipo polígono. Devuelve un valor numérico en las unidades del sistema de referencia (por ejemplo, metros cuadrados si está en proyección métrica).

Unión previa antes de la diferencia espacial

Los procesos de geoprocesamiento que se presentaron no son exclusivos de R ni del paquete **sf**, sino que forman parte del conjunto de herramientas fundamentales en cualquier Sistema de Información Geográfica (GIS). Estas operaciones son compartidas y utilizadas en otros lenguajes como Python (con librerías como **geopandas** y **shapely**) o en software de escritorio como QGIS o ArcGIS. La lógica subyacente es común a todos los entornos.

```
renabap_pba <-
  renabap %>% filter(provincia == 'Buenos Aires')
```

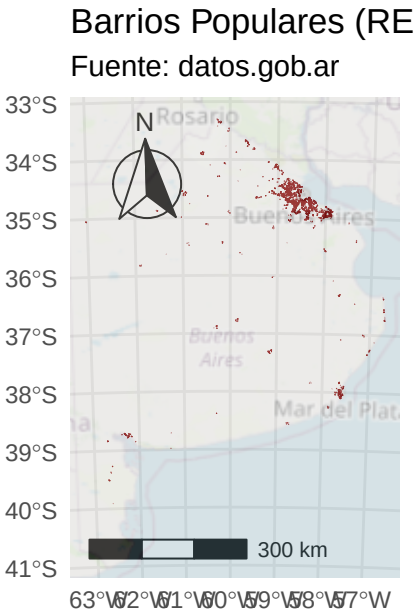
```
ggplot() +
  geom_sf(data = renabap_pba, fill = "tomato", color = "darkred", alpha = 0.5) +
  labs(title = "Barrios Populares (RENABAP)",
       subtitle = "Fuente: datos.gob.ar",
       caption = "Visualización con geom_sf()")+
  annotation_scale(location = "bl", width_hint = 0.5) +
  annotation_north_arrow(location = "tl", which_north = "true",
                         style = north_arrow_fancy_orienteering)+
  annotation_map_tile(alpha = 0.3)+
  theme_minimal()
```

```
Zoom: 5

Fetching 4 missing tiles

  |
  |                                                                      |   0%
  |
  |==================                                                    |  25%
  |
  |==================================                                    |  50%
  |
  |====================================================                  |  75%
  |
  |======================================================================| 100%
```

```
...complete!
```



Visualización con geom_sf()