



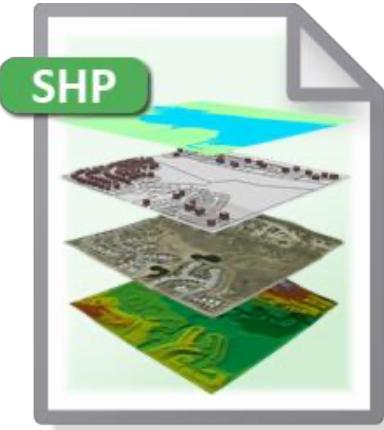
INSP/ESPM
ESCUELA DE SALUD
PÚBLICA DE MÉXICO

Despliegue de capas de información

José Luis Texcalac Sangrador

Procesos básicos de medición ambiental





- ▼  Municipios
 -  02mun.cpg
 -  02mun.dbf
 -  02mun.prj
 -  02mun.shp
 -  02mun.shx

Shapefile

Un shapefile se compone de varios Archivos.

Los mínimo son tres:

- El .shp** Almacena las entidades geométricas.
- El .shx** Almacena el índice de las entidades geométricas.
- El .dbf** Es la base de datos, en formato dBASE

También pueden tener un .prj, .sbn, .sbx, .fbn, .fbx .ain, .aih, .shp.xml.



TIPO DE DATOS QUE ALMACENA.

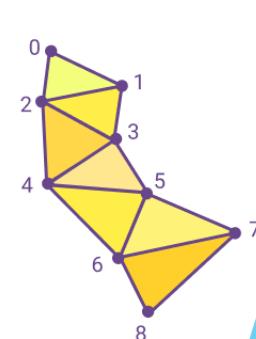


Punto



Polígono

MultiPach



MultiPunto



Es el formato más extendido y popular entre la comunidad GIS.



Importando datos con atributos espaciales en **R**

```
library(sf)
```

```
objeto <-  
  st_read("ruta_archivo.shp",  
          stringsAsFactors = FALSE,  
          options = "ENCODING = WINDOWS-1252")
```

Alias:

```
read_sf("ruta_archivo.shp")
```



Mapa básico con ggplot

```
ggplot( ) +  
  geom_sf( ) +  
  theme( )
```

```
geom_sf(  
  mapping = aes(),  
  data = NULL,  
  stat = "sf",  
  position = "identity",  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE,  
  ...  
)
```



Exportar shapefile

```
st_write(objeto, "ruta/nombre_archivo.shp")
```

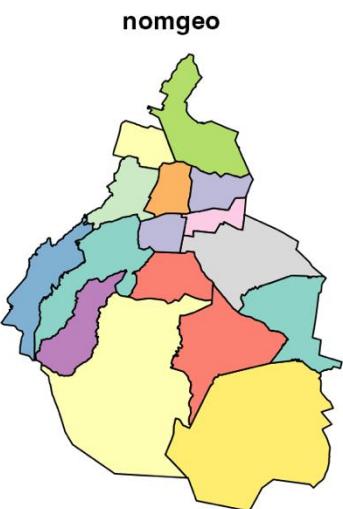
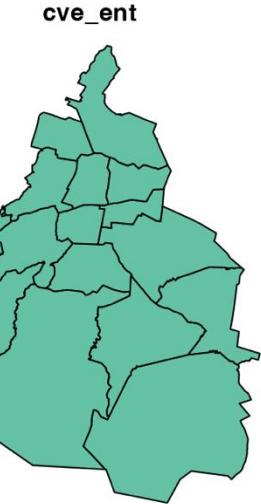


Importar capa de información

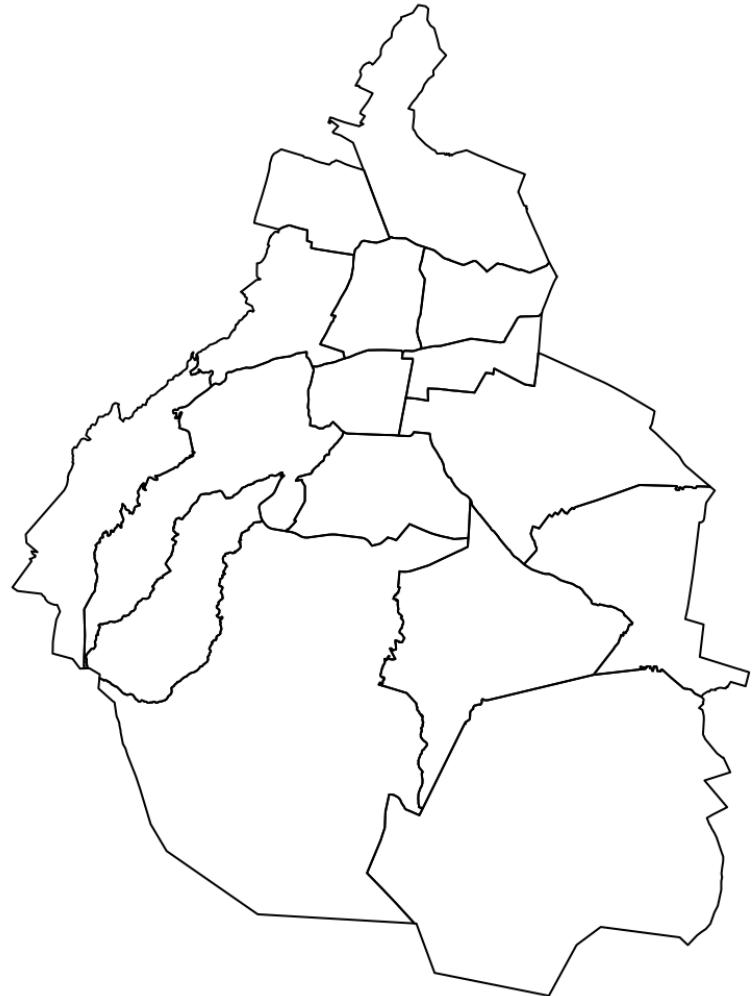
```
mun <-
  st_read("./shapefiles/cdmx_mun.shp") %>%
  clean_names() %>%
  print()
```



plot(mun)



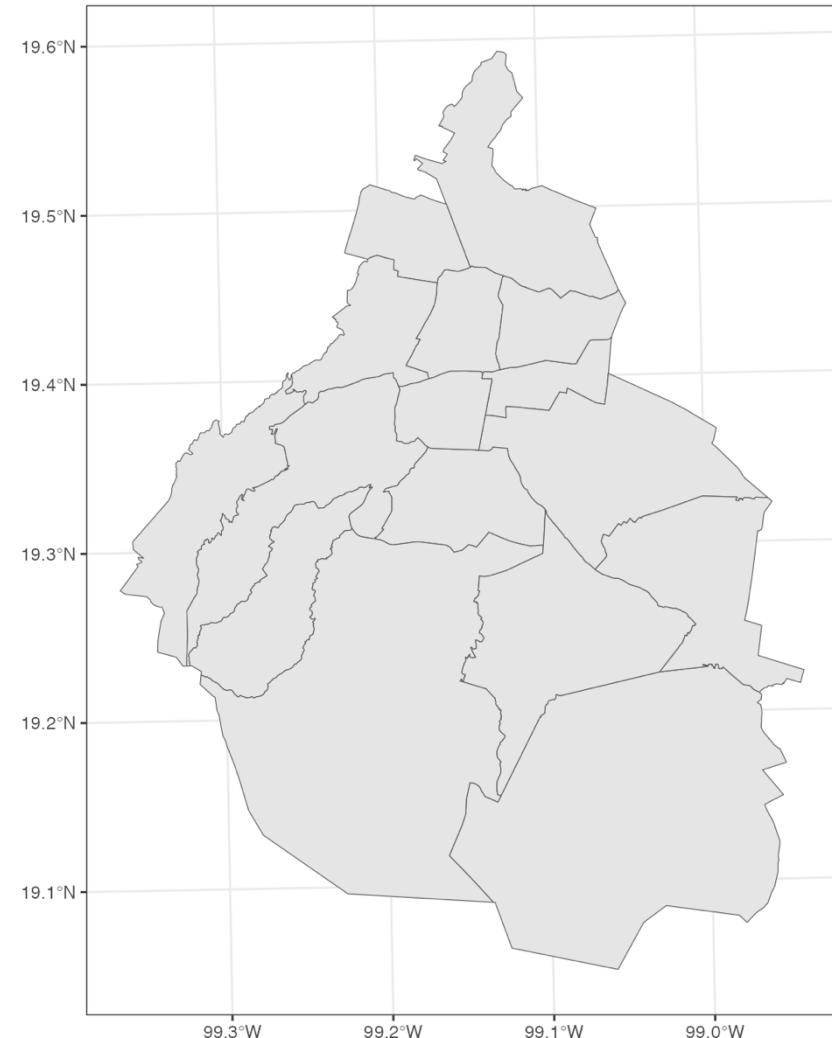
plot(st_geometry(mun))





Visualizar capa de información con ggplot

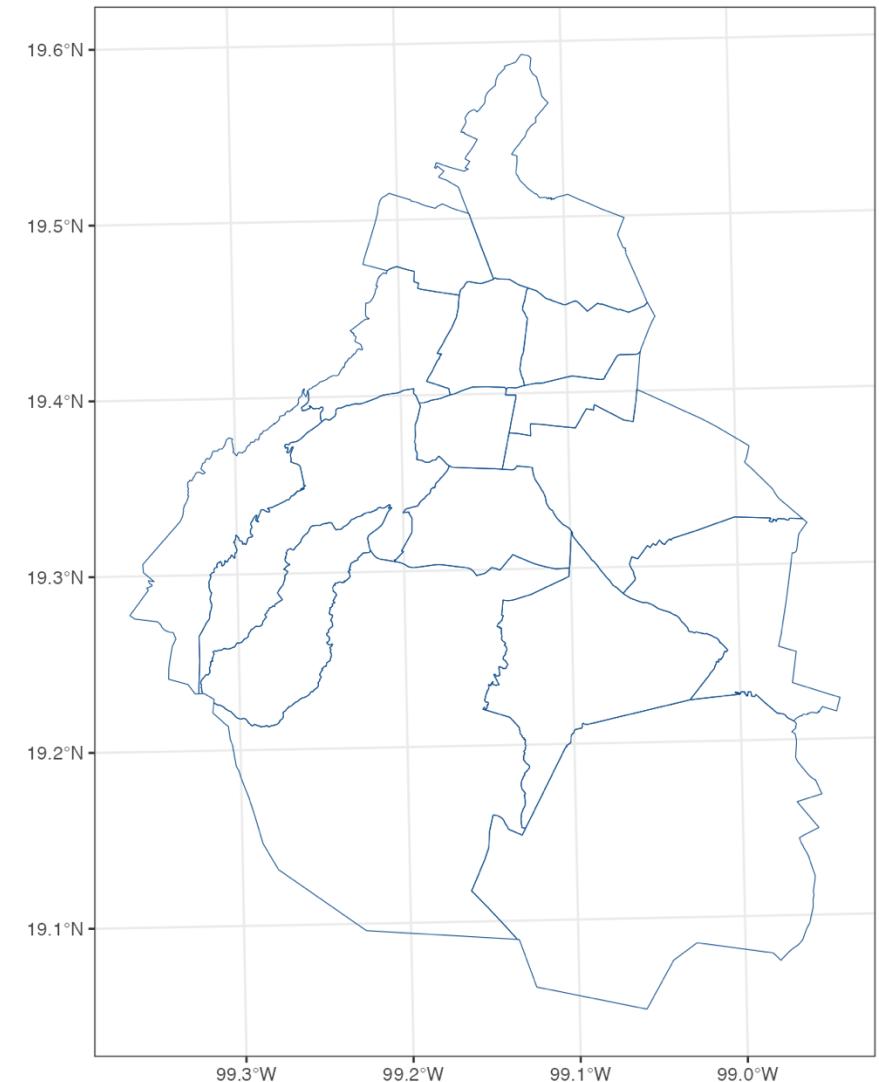
```
ggplot(mun) +  
  geom_sf(data = mun) +  
  theme_bw()
```





Visualizar capa de información con ggplot

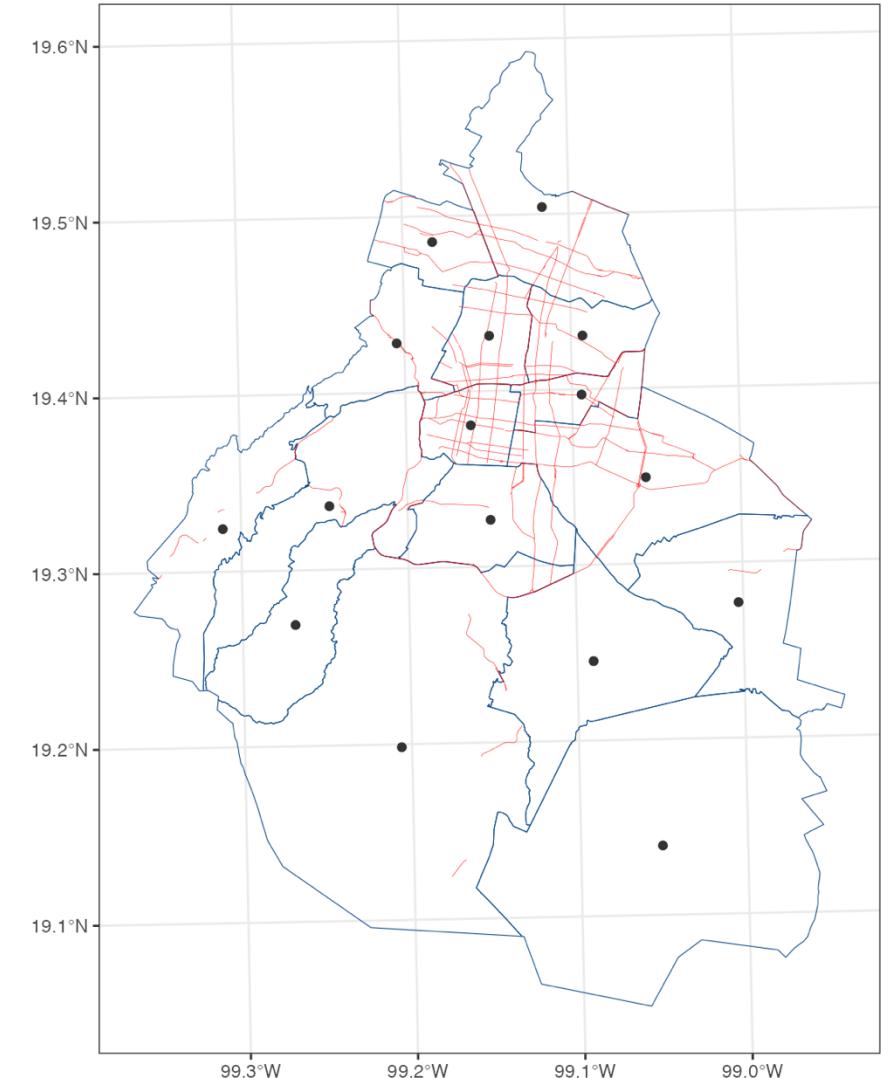
```
ggplot() +  
  geom_sf(data = mun,  
          color = "dodgerblue4",  
          fill = "transparent") +  
  theme_bw()
```





Cartografía de puntos, líneas y polígonos

```
ggplot() +  
  geom_sf(data = mun, color = "dodgerblue4", fill = "transparent") +  
  geom_sf(data = vial, color = "red", linewidth = 0.1) +  
  geom_sf(data = cent, color = "grey20") +  
  theme_bw()
```

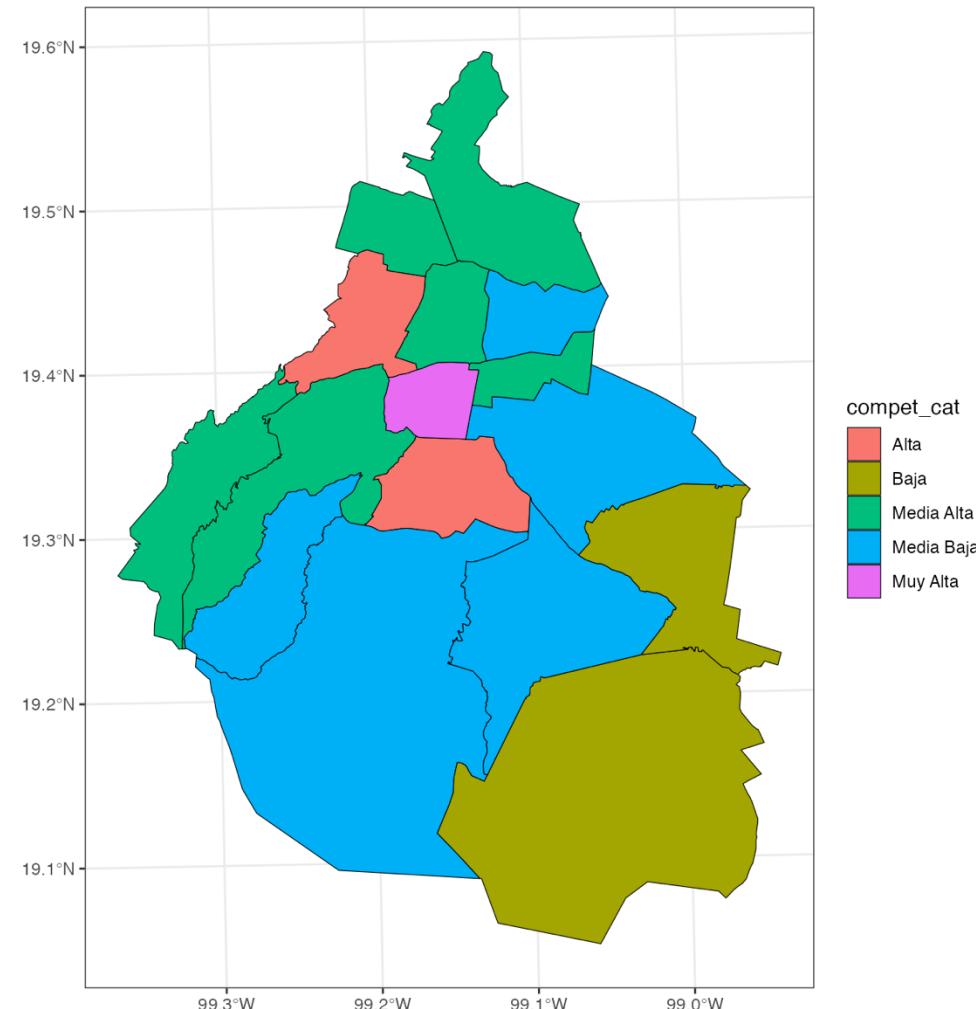




Cartografía temática de datos categóricos

cvegeo	compet_cat	compet_cont
<chr>	<chr>	<dbl>
09002	Media Alta	55.1
09003	Alta	64.7
09004	Media Alta	52.2
09005	Media Alta	50
09006	Media Alta	51
09007	Media Baja	41.5
09008	Media Baja	41.1
09009	Baja	30.1
09010	Media Alta	52.7
09011	Baja	33.6
09012	Media Baja	49.3
09013	Media Baja	36.4
09014	Muy Alta	78.3
09015	Media Alta	50.8
09016	Alta	74.4
09017	Media Baja	37.8

```
ggplot() +  
  geom_sf(data = cdmx_imco,  
          color = "black",  
          aes(fill = compet_cat) +  
  theme_bw()
```

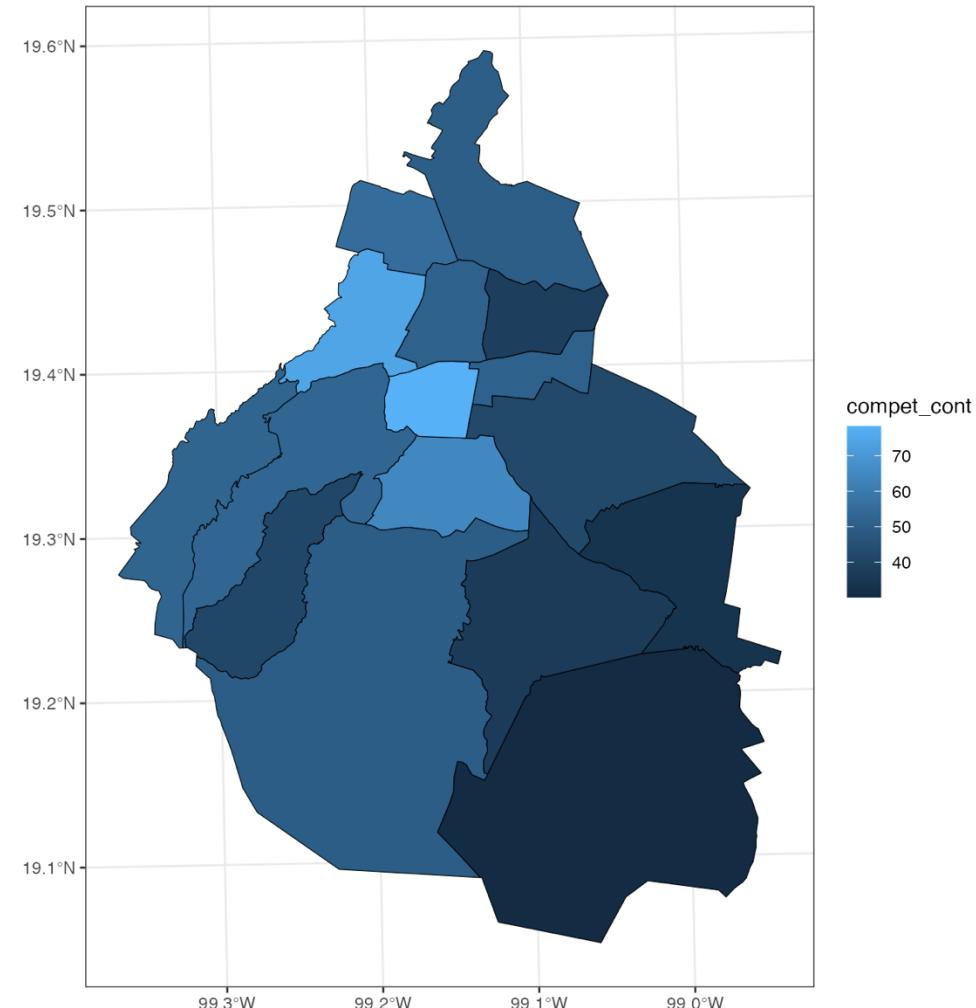




Cartografía temática de datos continuos

cvegeo	compet_cat	compet_cont
<chr>	<chr>	<dbl>
09002	Media Alta	55.1
09003	Alta	64.7
09004	Media Alta	52.2
09005	Media Alta	50
09006	Media Alta	51
09007	Media Baja	41.5
09008	Media Baja	41.1
09009	Baja	30.1
09010	Media Alta	52.7
09011	Baja	33.6
09012	Media Baja	49.3
09013	Media Baja	36.4
09014	Muy Alta	78.3
09015	Media Alta	50.8
09016	Alta	74.4
09017	Media Baja	37.8

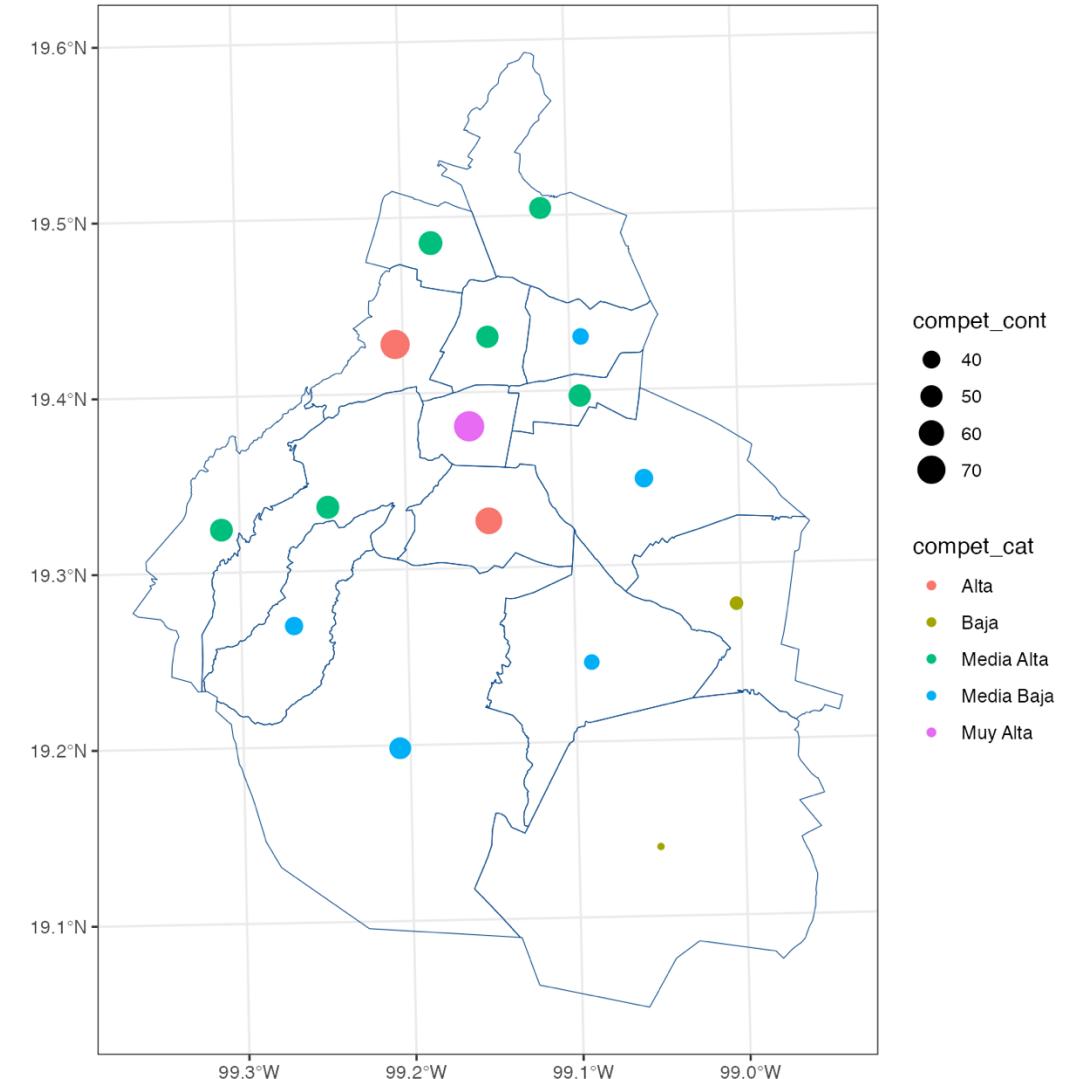
```
ggplot() +  
  geom_sf(data = cdmx_imco,  
          color = "black",  
          aes(fill = compet_cont) +  
  theme_bw()
```



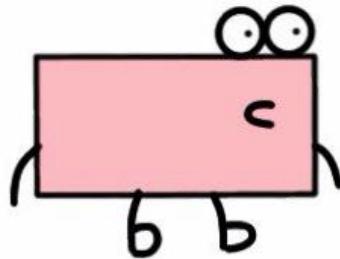


Cartografía temática de dos variables

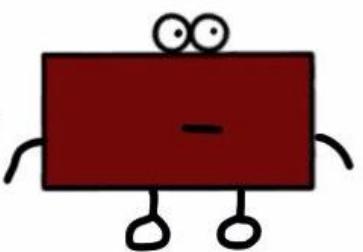
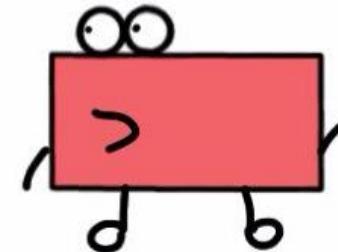
```
ggplot() +  
  geom_sf(data = mun,  
          color = "dodgerblue4",  
          fill = "transparent") +  
  geom_sf(data = cdmx_imco_c,  
          aes(size = compet_cont,  
              color = compet_cat)) +  
  theme_bw()
```



You're pretty hot



You should see
the cell next to me



Cartografía temática

José Luis Texcalac Sangrador

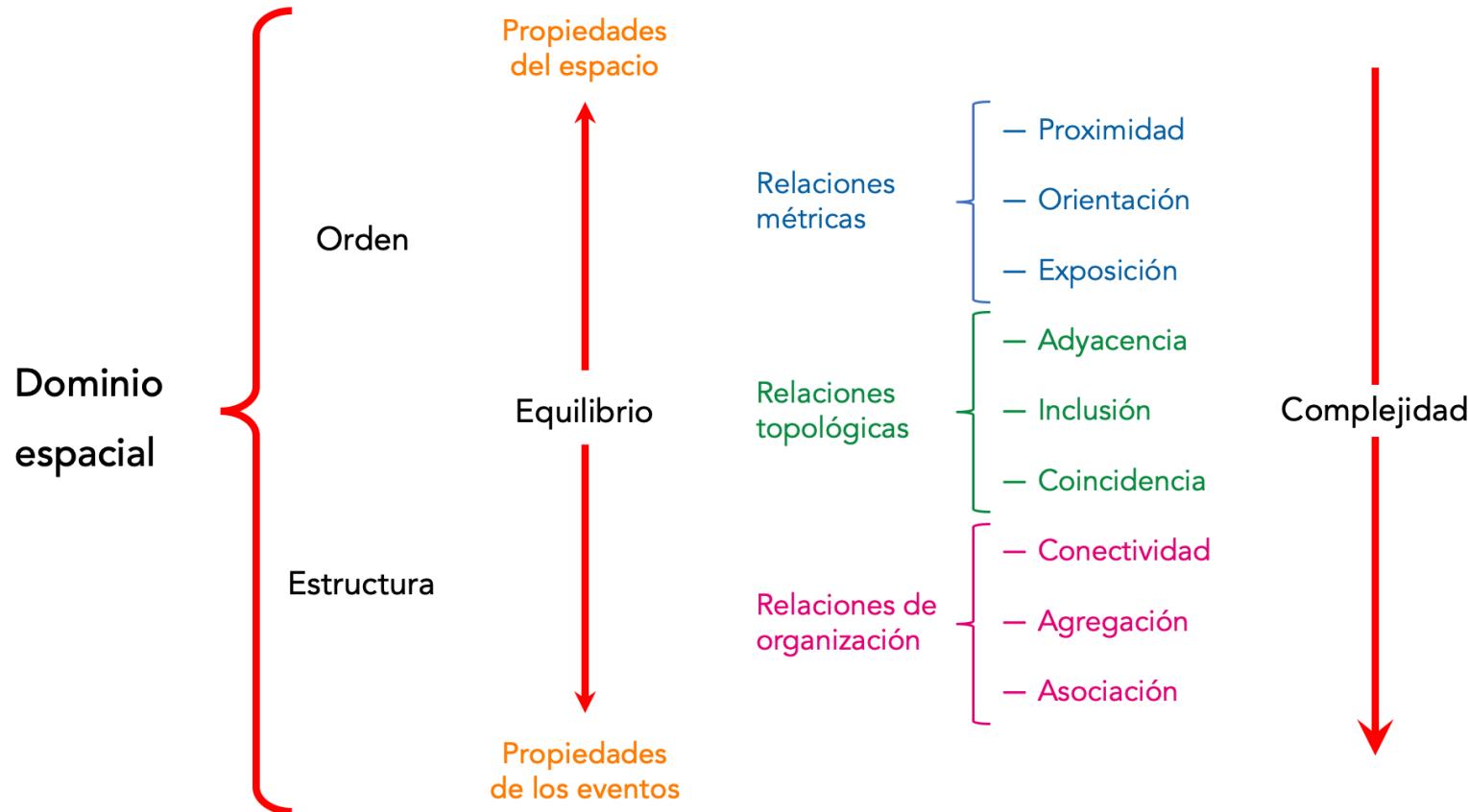
Procesos básicos de medición ambiental

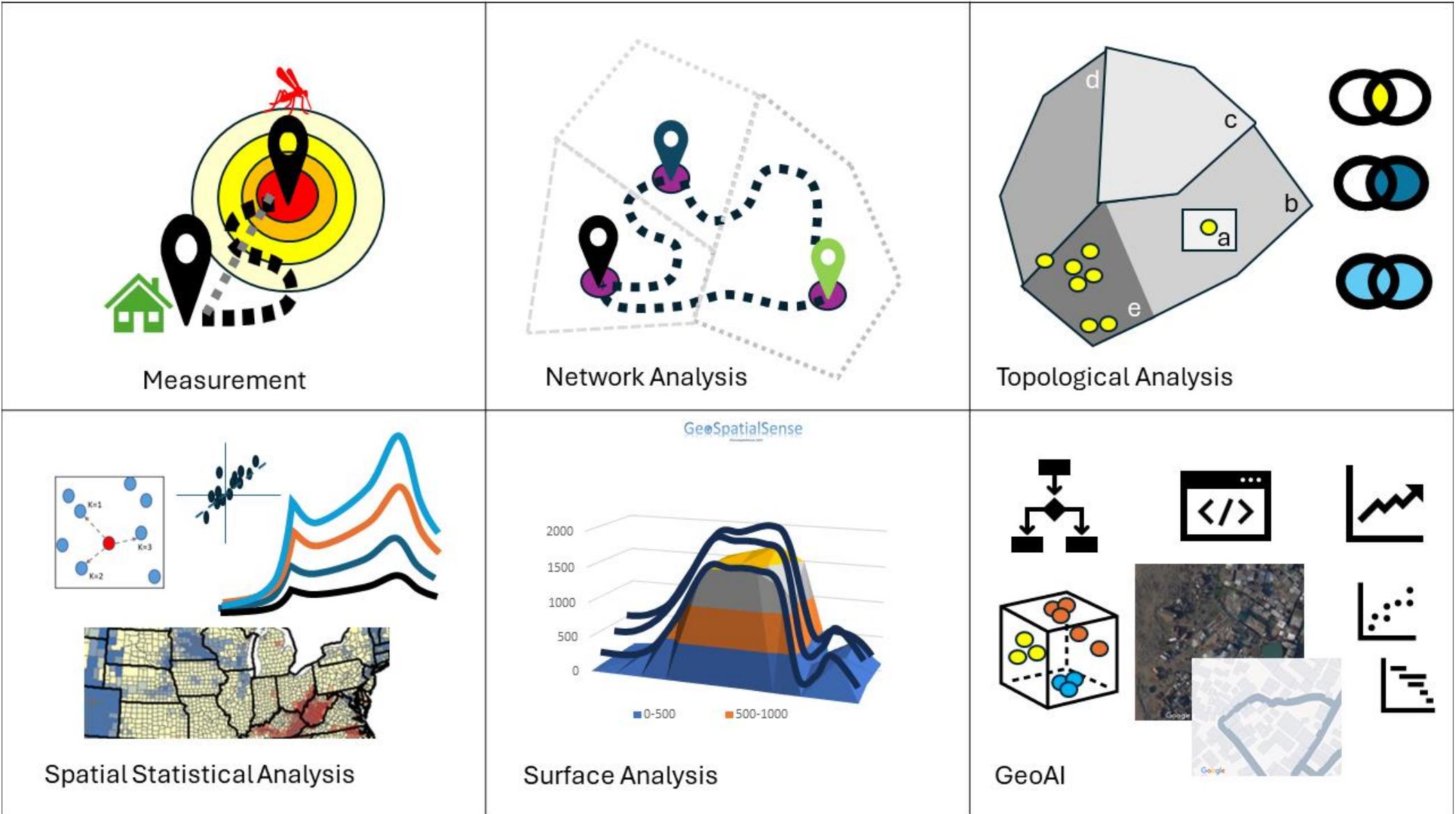


Relaciones espaciales

- Son conceptos que surgen de la interacción entre el espacio y los eventos que en él ocurren.
- Se resumen en 9 tipos y tres grandes grupos.
 - Predominio de las propiedades del espacio
 - Equilibrio entre las propiedades del espacio y los eventos que en él ocurren.
 - Patrones espaciales definidos por las propiedades del espacio y los eventos que en él ocurren.

Relaciones espaciales







Equals A is the same as B	Two overlapping circles, one green and one blue, labeled 'A' and 'B' respectively, representing sets that are equal.
Touches A touches B	A green circle labeled 'A' touching a blue circle labeled 'B' at a single point, representing sets that touch.
Overlaps A and B have multiple points in common	Two overlapping circles, one green and one blue, labeled 'A' and 'B' respectively, representing sets that overlap.
Contains A contains B	A large green circle labeled 'A' containing a smaller blue circle labeled 'B' entirely within its boundary, representing set A containing set B.
Disjoint A shares nothing with B	Two separate circles, one green and one blue, labeled 'A' and 'B' respectively, representing sets that are disjoint.
Covers A covers B (or vice versa)	A large green circle labeled 'A' completely enclosing a smaller blue circle labeled 'B', representing set A covering set B.
Crosses A and B have at least one point in common	Two blue line segments, one vertical and one L-shaped, representing sets A and B that cross each other.



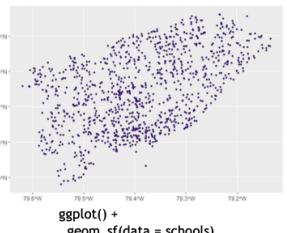
Spatial manipulation with sf: : CHEAT SHEET

The sf package provides a set of tools for working with geospatial vectors, i.e. points, lines, polygons, etc.



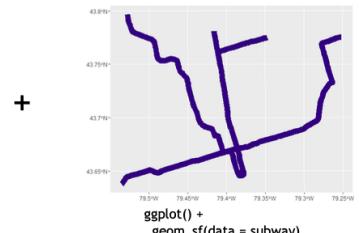
Geometric confirmation

- ☒ ○ st_contains(x, y, ...) Identifies if y is within x (i.e. point within polygon)
- ☒ ○ st_covered_by(x, y, ...) Identifies if x is completely within y (i.e. polygon completely within polygon)
- ☒ ○ st_covers(x, y, ...) Identifies if any point from x is outside of y (i.e. polygon outside polygon)
- ☒ ○ st_crosses(x, y, ...) Identifies if any geometry of x have commonalities with y
- ☒ ○ st_disjoint(x, y, ...) Identifies when geometries from x do not share space with y
- ☒ ○ st_equals(x, y, ...) Identifies if x and y share the same geometry
- ☒ ○ st_intersects(x, y, ...) Identifies if x and y geometry share any space
- ☒ ○ st_overlaps(x, y, ...) Identifies if geometries of x and y share space, are of the same dimension, but are not completely contained by each other
- ☒ ○ st_touches(x, y, ...) Identifies if geometries of x and y share a common point but their interiors do not intersect
- ☒ ○ st_within(x, y, ...) Identifies if x is in a specified distance to y



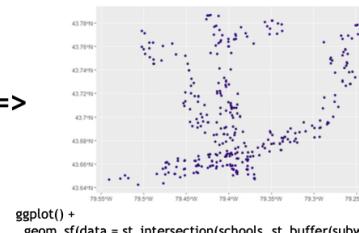
Geometric operations

- ☒ ○ st_boundary(x) Creates a polygon that encompasses the full extent of the geometry
- ☒ ○ st_buffer(x, dist, nQuadSegs) Creates a polygon covering all points of the geometry within a given distance
- ☒ ○ st_centroid(x, ..., of_largest_polygon) Creates a point at the geometric centre of the geometry
- ☒ ➔ st_convex_hull(x) Creates geometry that represents the minimum convex geometry of x
- ☒ ➔ st_line_merge(x) Creates linestring geometry from sewing multi linestring geometry together
- ☒ ➔ st_node(x) Creates nodes on overlapping geometry where nodes do not exist
- ☒ ○ st_point_on_surface(x) Creates a point that is guaranteed to fall on the surface of the geometry
- ☒ ➔ st_polygonize(x) Creates polygon geometry from linestring geometry
- ☒ ➔ st_segmentize(x, dfMaxLength, ...) Creates linestring geometry from x based on a specified length
- ☒ ➔ st_simplify(x, preserveTopology, dTolerance) Creates a simplified version of the geometry based on a specified tolerance



Geometry creation

- ☒ ➔ st_triangulate(x, dTolerance, bOnlyEdges) Creates polygon geometry as triangles from point geometry
- ☒ ➔ st_voronoi(x, envelope, dTolerance, bOnlyEdges) Creates polygon geometry covering the envelope of x, with x at the centre of the geometry
- ☒ ○ st_point(x, c(numeric vector), dim = "XYZ") Creating point geometry from numeric values
- ☒ ○ st_multipoint(x = matrix(numeric values in rows), dim = "XYZ") Creating multi point geometry from numeric values
- ☒ ○ st_linestring(x = matrix(numeric values in rows), dim = "XYZ") Creating linestring geometry from numeric values
- ☒ ○ st_multilinestring(x = list(numeric matrices in rows), dim = "XYZ") Creating multi linestring geometry from numeric values
- ☒ ○ st_polygon(x = list(numeric matrices in rows), dim = "XYZ") Creating polygon geometry from numeric values
- ☒ ○ st_multipolygon(x = list(numeric matrices in rows), dim = "XYZ") Creating multi polygon geometry from numeric values





Spatial manipulation with sf: : CHEAT SHEET

The sf package provides a set of tools for working with geospatial vectors, i.e. points, lines, polygons, etc.



Geometry operations

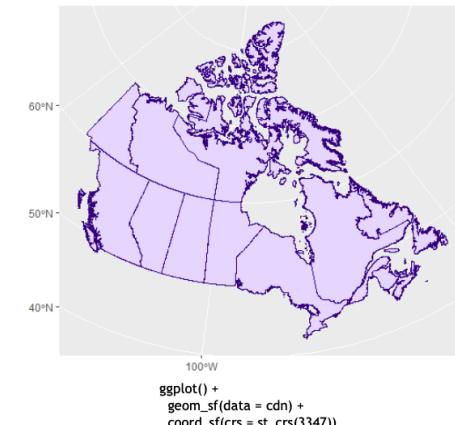
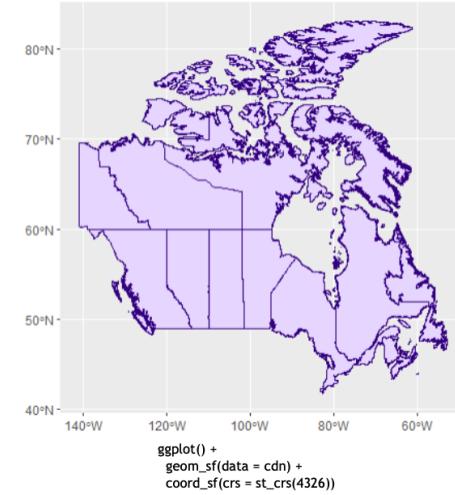
- ⦿ ⊕ `st_contains(x, y, ...)` Identifies if y is within x (i.e. point within polygon)
- ⦿ ⊞ `st_crop(x, y, ..., xmin, ymin, xmax, ymax)` Creates geometry of x that intersects a specified rectangle
- ⦿ ⊞ `st_difference(x, y)` Creates geometry from x that does not intersect with y
- ⦿ ⊞ `st_intersection(x, y)` Creates geometry of the shared portion of x and y
- ⦿ ⊞ `st_sym_difference(x, y)` Creates geometry representing portions of x and y that do not intersect
- ⦿ ⊞ `st_snap(x, y, tolerance)` Snap nodes from geometry x to geometry y
- ⦿ ⊞ `st_union(x, y, ..., by_feature)` Creates multiple geometries into a single geometry, consisting of all geometry elements

Geometric measurement

- ⦿ `st_area(x)` Calculate the surface area of a polygon geometry based on the current coordinate reference system
- ⦿ `st_distance(x, y, ..., dist_fun, by_element, which)` Calculates the 2D distance between x and y based on the current coordinate system
- ⦿ `st_length(x)` Calculates the 2D length of a geometry based on the current coordinate system

Misc operations

- ⦿ `st_as_sf(x, ...)` Create a sf object from a non-geospatial tabular data frame
- ⦿ `st_cast(x, to, ...)` Change x geometry to a different geometry type
- ⦿ `st_coordinates(x, ...)` Creates a matrix of coordinate values from x
- ⦿ `st_crs(x, ...)` Identifies the coordinate reference system of x
- ⦿ `st_join(x, y, join, FUN, suffix, ...)` Performs a spatial left or inner join between x and y
- ⦿ `st_make_grid(x, cellsize, offset, n, crs, what)` Creates rectangular grid geometry over the bounding box of x
- ⦿ `st_nearest_feature(x, y)` Creates an index of the closest feature between x and y
- ⦿ `st_nearest_points(x, y, ...)` Returns the closest point between x and y
- ⦿ `st_read(dsn, layer, ...)` Read file or database vector dataset as a sf object
- ⦿ `st_transform(x, crs, ...)` Convert coordinates of x to a different coordinate reference system





Proyecciones en R

Siempre nuestro planteamiento inicial debería ser
¿son iguales las proyecciones de las capas que utilizaré?

```
> st_crs(capa)
```

Me indica la proyección de mi capa

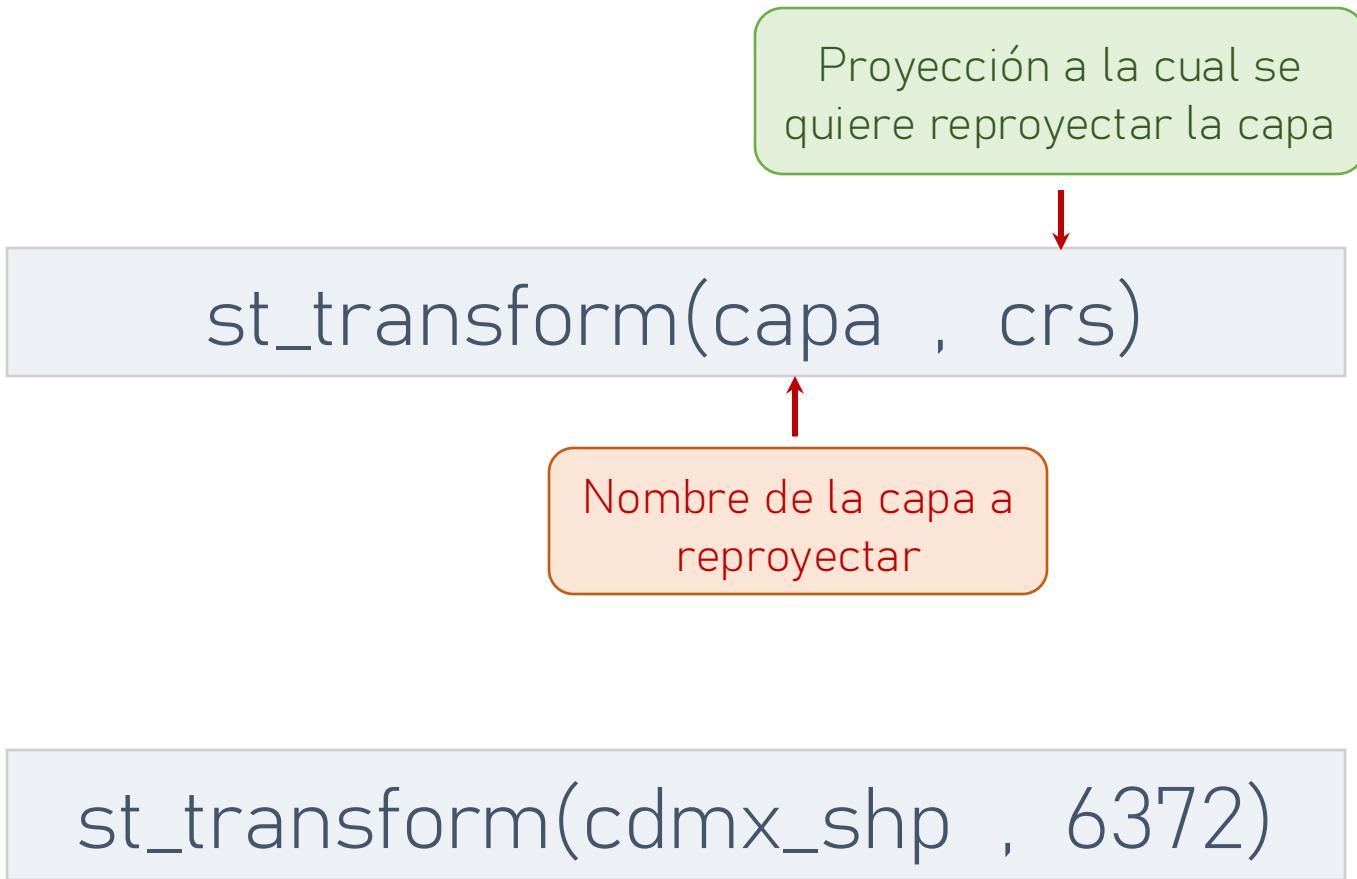
```
identical(st_crs(capa1), st_crs(capa2))
```

Identifico si las proyecciones de dos capas son similares

Todo análisis espacial debe realizarse con
capas que tienen la misma proyección.



Reproyectando una capa en R



INEGI: 6372
WGS84: 4326



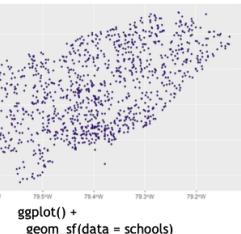
Spatial manipulation with sf: : CHEAT SHEET

The sf package provides a set of tools for working with geospatial vectors, i.e. points, lines, polygons, etc.



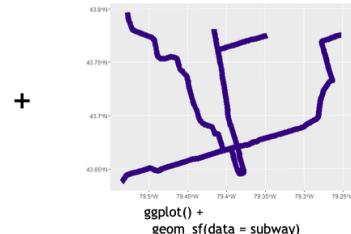
Geometric confirmation

- ▢ ⊕ st_contains(x, y, ...) Identifies if y is within x (i.e. point within polygon)
- ▢ ⊕ st_covered_by(x, y, ...) Identifies if x is completely within y (i.e. polygon completely within polygon)
- ▢ ⊕ st_covers(x, y, ...) Identifies if any point from x is outside of y (i.e. polygon outside polygon)
- ▢ ⊕ st_crosses(x, y, ...) Identifies if any geometry of x have commonalities with y
- ▢ ⊕ st_disjoint(x, y, ...) Identifies when geometries from x do not share space with y
- ▢ ⊕ st_equals(x, y, ...) Identifies if x and y share the same geometry
- ▢ ⊕ st_intersects(x, y, ...) Identifies if x and y geometry share any space
- ▢ ⊕ st_overlaps(x, y, ...) Identifies if geometries of x and y share space, are of the same dimension, but are not completely contained by each other
- ▢ ⊕ st_touches(x, y, ...) Identifies if geometries of x and y share a common point but their interiors do not intersect
- ▢ ⊕ st_within(x, y, ...) Identifies if x is in a specified distance to y



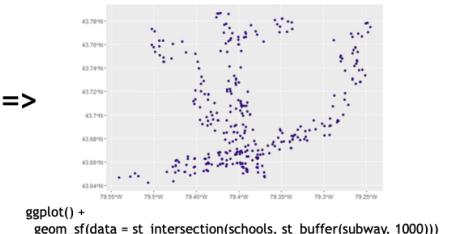
Geometric operations

- ▢ ⊕ st_boundary(x) Creates a polygon that encompasses the full extent of the geometry
- ▢ ⊕ st_buffer(x, dist, nQuadSegs) Creates a polygon covering all points of the geometry within a given distance
- ▢ ⊕ st_centroid(x, ..., of_largest_polygon) Creates a point at the geometric centre of the geometry
- ▢ ⊞ st_convex_hull(x) Creates geometry that represents the minimum convex geometry of x
- ▢ ⊞ st_line_merge(x) Creates linestring geometry from sewing multi linestring geometry together
- ▢ ⊞ st_node(x) Creates nodes on overlapping geometry where nodes do not exist
- ▢ ⊞ st_point_on_surface(x) Creates a point that is guaranteed to fall on the surface of the geometry
- ▢ ⊞ st_polygonize(x) Creates polygon geometry from linestring geometry
- ▢ ⊞ st_segmentize(x, dfMaxLength, ...) Creates linestring geometry from x based on a specified length
- ▢ ⊞ st_simplify(x, preserveTopology, dTolerance) Creates a simplified version of the geometry based on a specified tolerance



Geometry creation

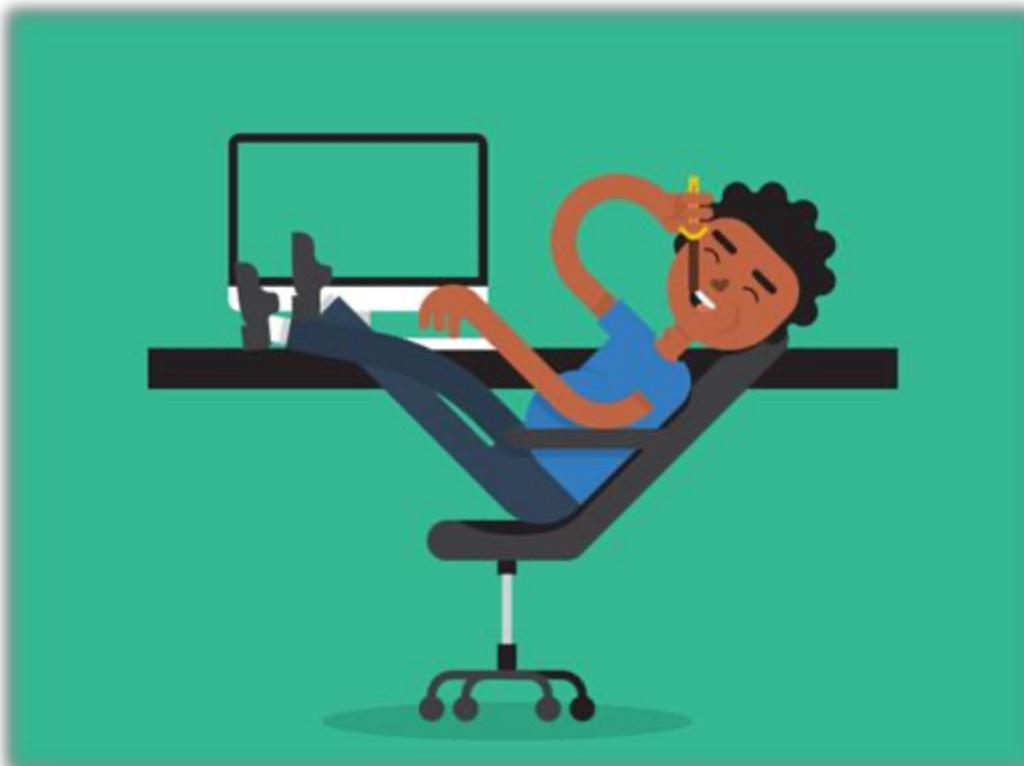
- ▢ ⇒ ⊕ st_triangulate(x, dTolerance, bOnlyEdges) Creates polygon geometry as triangles from point geometry
- ▢ ⇒ ⊕ st_voronoi(x, envelope, dTolerance, bOnlyEdges) Creates polygon geometry covering the envelope of x, with x at the centre of the geometry
- ▢ ⊕ st_point(x, c(numeric vector), dim = "XYZ") Creating point geometry from numeric values
- ▢ ⊕ st_multipoint(x = matrix(numeric values in rows), dim = "XYZ") Creating multi point geometry from numeric values
- ▢ ⊕ st_linestring(x = matrix(numeric values in rows), dim = "XYZ") Creating linestring geometry from numeric values
- ▢ ⊕ st_multilinestring(x = list(numeric matrices in rows), dim = "XYZ") Creating multi linestring geometry from numeric values
- ▢ ⊕ st_polygon(x = list(numeric matrices in rows), dim = "XYZ") Creating polygon geometry from numeric values
- ▢ ⊕ st_multipolygon(x = list(numeric matrices in rows), dim = "XYZ") Creating multi polygon geometry from numeric values





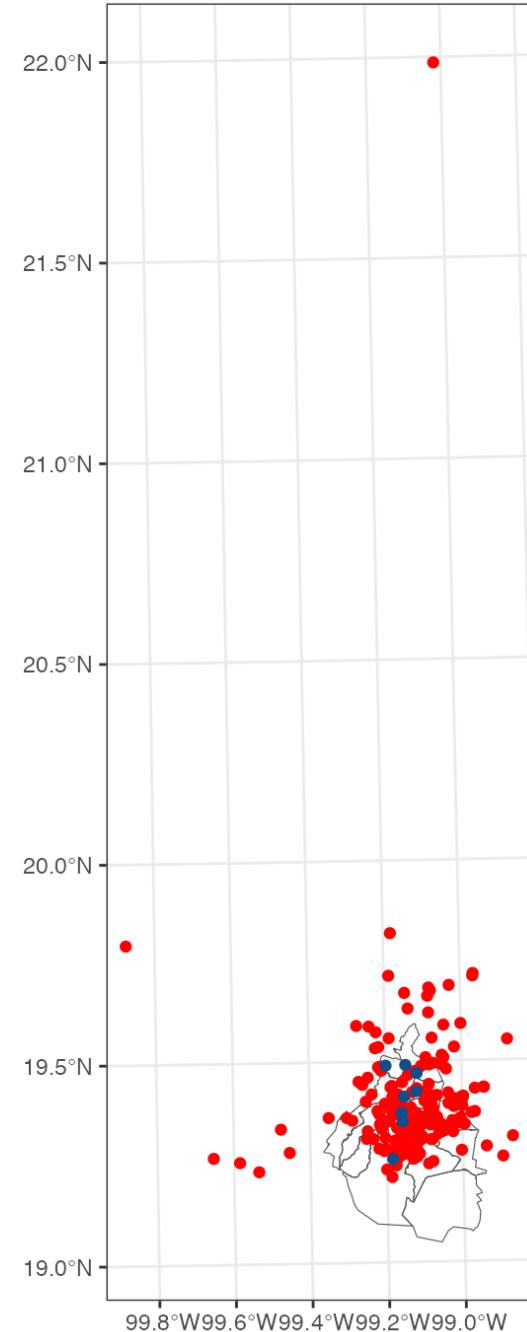
Su turno...

- Importe la capa de municipios de la CDMX que descargó de INEGI ([cdmx_mun](#))
- Importe la capa de sitios ([sitios](#))
- Importe la capa de monitores ([monit](#))
- ¿qué sigue?
- Genere un mapa con ambas capas





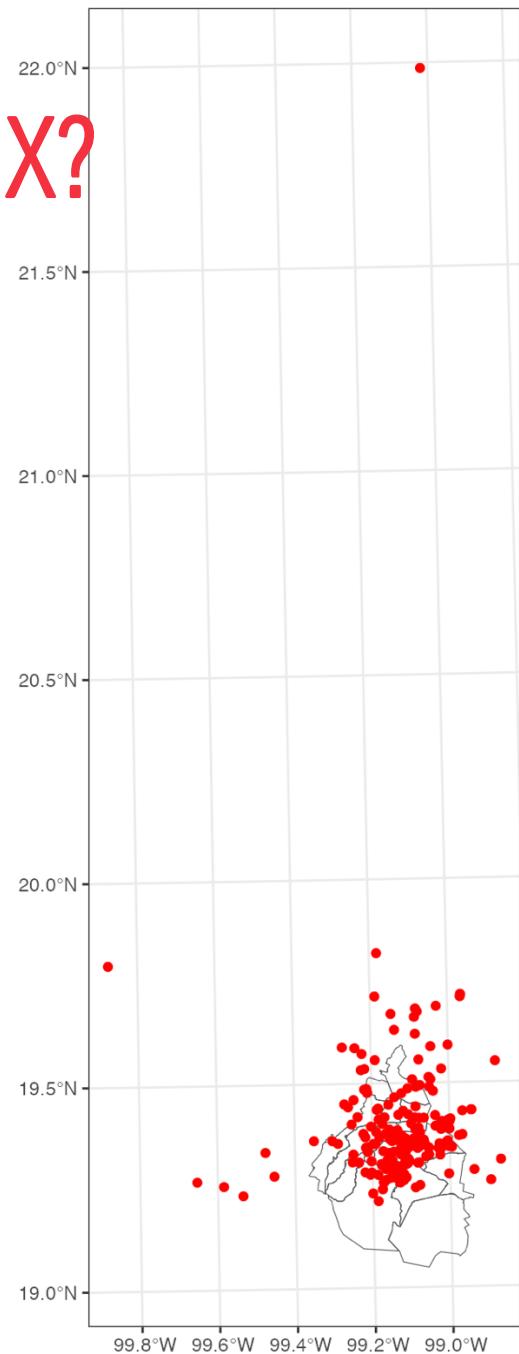
```
ggplot() +  
  geom_sf(data = cdmx_mun,  
          fill = "transparent") +  
  geom_sf(data = sitios,  
          color = "red") +  
  geom_sf(data = monit,  
          color = "dodgerblue4") +  
  theme_bw()
```





¿Qué folios están dentro de CDMX?

```
sitios_cdmx <-  
  st_filter(sitios, cdmx) %>%  
  print()
```

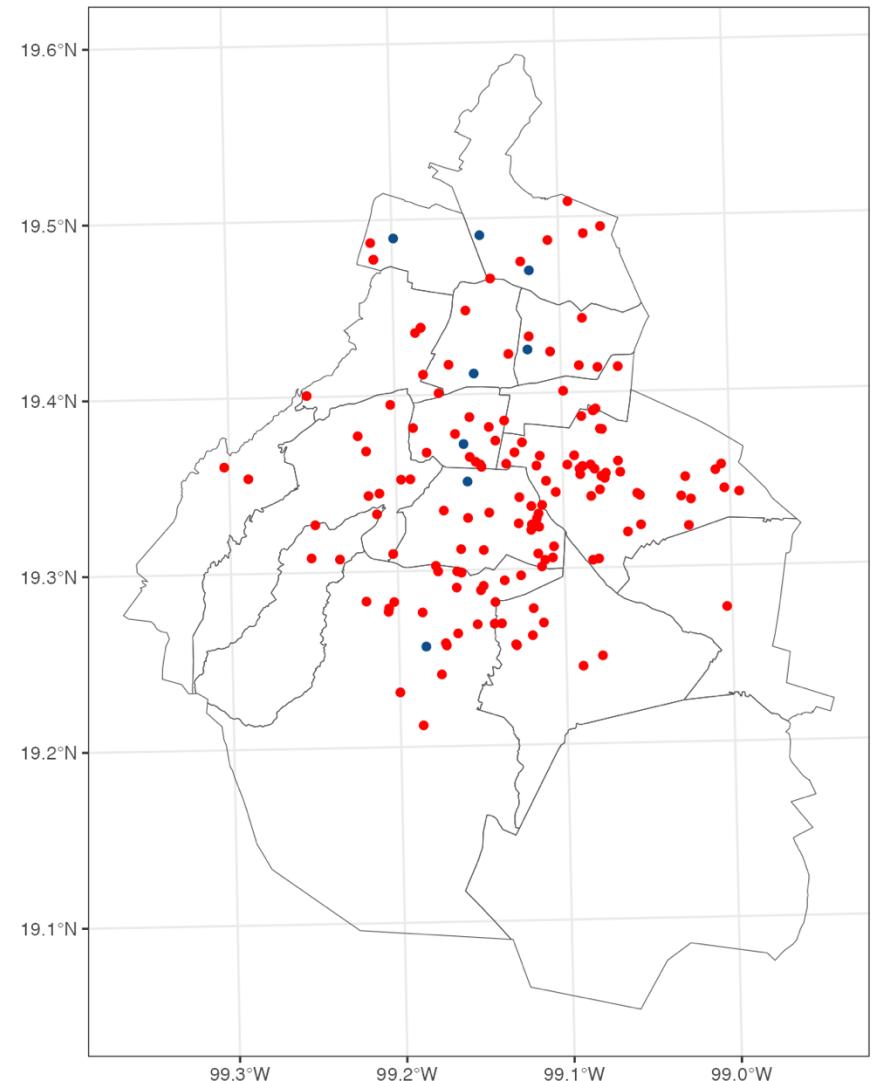




¿Qué folios están dentro de CDMX?

```
sitios_cdmx <-  
  st_filter(sitios, cdmx) %>%  
  print()
```

```
ggplot() +  
  geom_sf(data = cdmx_mun, fill = "transparent") +  
  geom_sf(data = sitios_cdmx, color = "red") +  
  geom_sf(data = monit, color = "dodgerblue4") +  
  theme_bw()
```

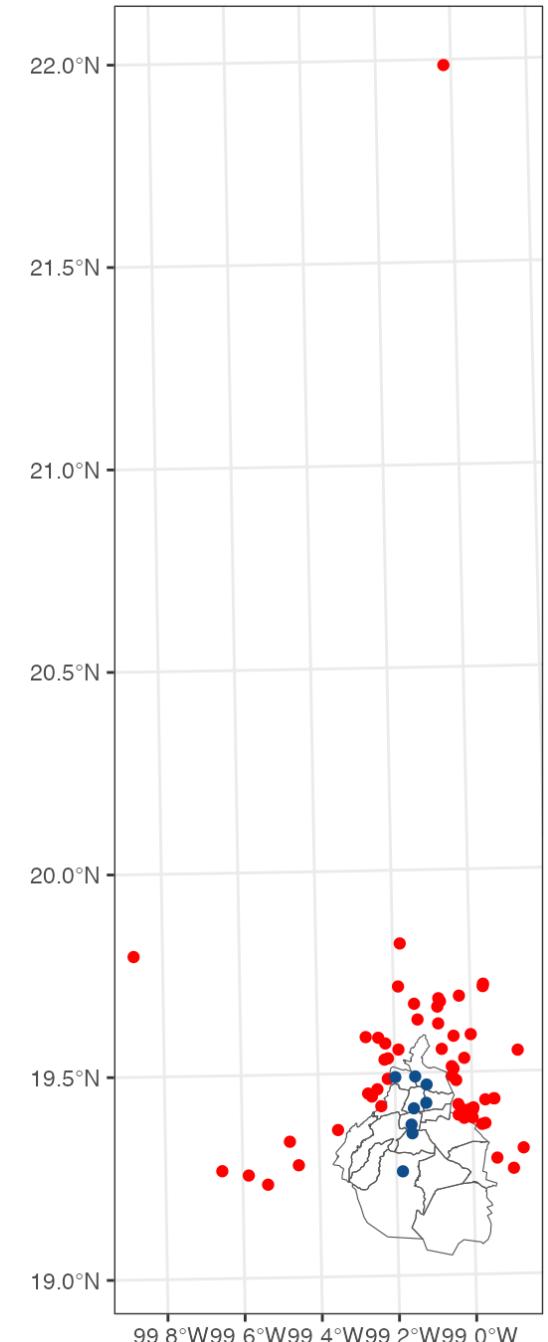




¿Qué folios están fuera de CDMX?

```
sitios_no_cdmx <-  
  st_filter(sitios, cdmx_ent, .predicate = st_disjoint) %>%  
  print()
```

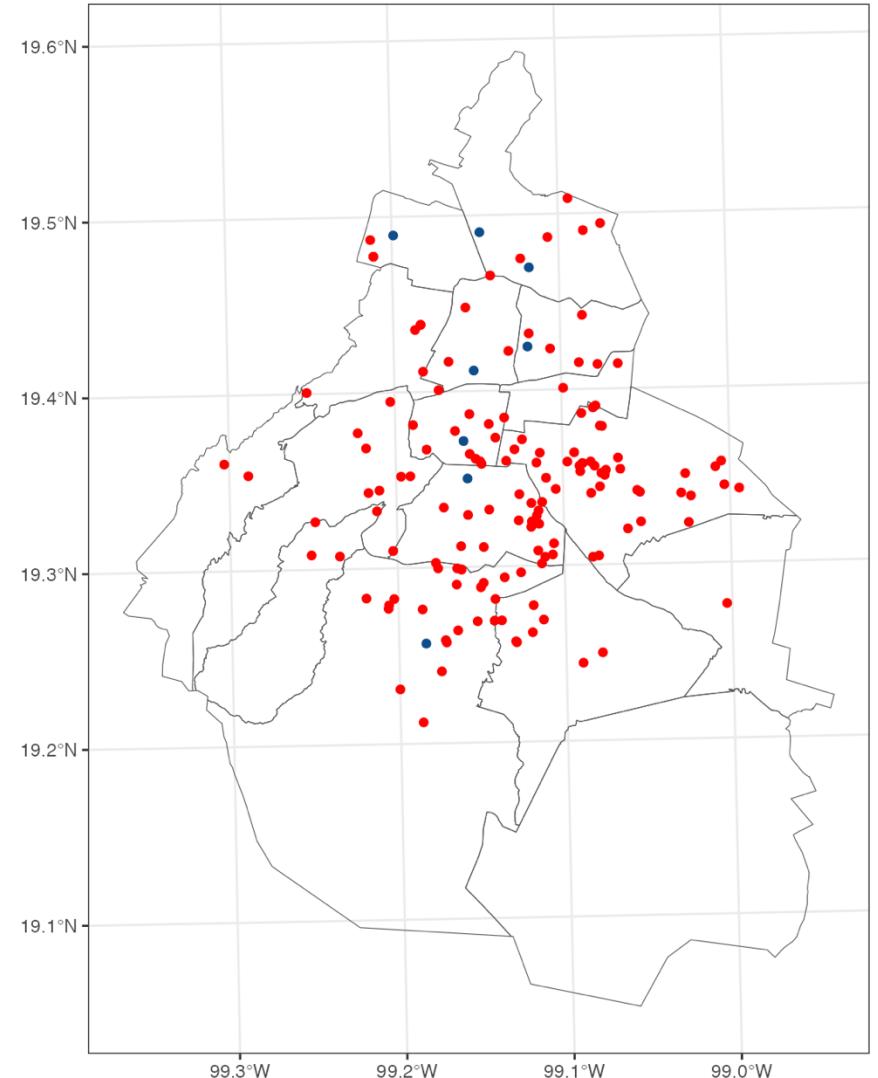
```
ggplot() +  
  geom_sf(data = cdmx_ent, fill = "transparent") +  
  geom_sf(data = sitios_no_cdmx, color = "red") +  
  geom_sf(data = monit, color = "dodgerblue4") +  
  theme_bw()
```





¿A qué alcaldía corresponde cada folio?

```
st_join(sitios_cdmx, cdmx_mun, join = st_within)
```



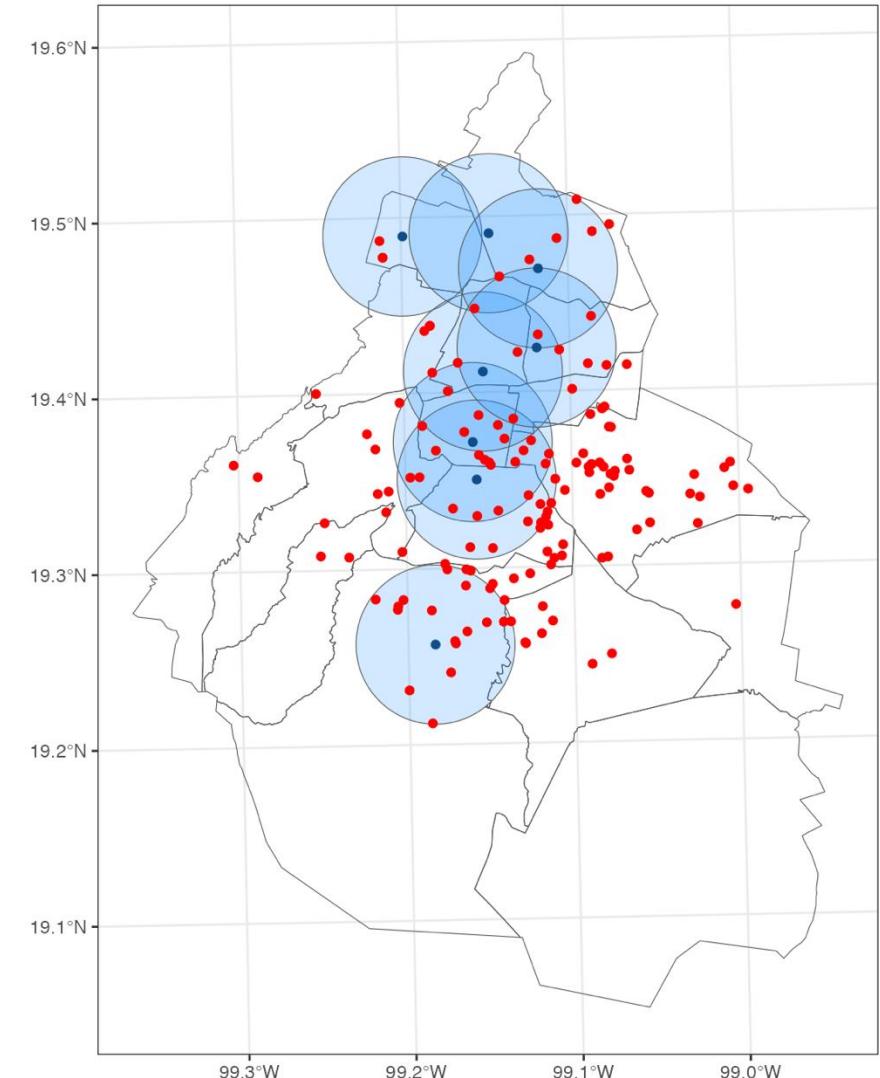


Área de influencia de un punto (buffer)

```
buf_monit <-
  st_buffer(monit, 5000, nQuadSegs = 30) %>%
  print()
```

¿A qué buffer corresponde cada sitio?

¿cuántos sitios quedaron fuera de buffers?





¿En qué buffer o buffers cae cada folio?

```
st_join(pts_cdmx,  
       buf_monit,  
       join = st_within)
```

Simple feature collection with 183 features and 6 fields

Geometry type: POINT

Dimension: XY

Bounding box: xmin: 2782545 ymin: 804611.1 xmax: 2814894 ymax: 837532.4

Projected CRS: Mexico ITRF2008 / LCC

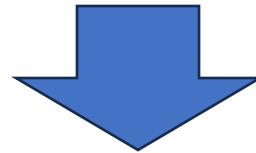
First 10 features:

	folio	cvegeo	cve_ent	cve_mun	nomgeo	site	geometry
1	1754	09005	09	005	Gustavo A. Madero	LVI	POINT (2805067 835529.4)
2	2722	09012	09	012		Tlalpan	<NA> POINT (2799581 812358.5)
3	8758	09003	09	003		Coyoacán	COY POINT (2801880 817228)
4	2192	09007	09	007		Iztapalapa	<NA> POINT (2806450 820158.9)
5	3161	09005	09	005	Gustavo A. Madero	IMP	POINT (2801141 833745)
5.1	3161	09005	09	005	Gustavo A. Madero	LVI	POINT (2801141 833745)
6	8830	09012	09	012		Tlalpan	<NA> POINT (2801194 814034.4)
7	1938	09007	09	007		Iztapalapa	<NA> POINT (2806145 819439)
8	8606	09007	09	007		Iztapalapa	<NA> POINT (2804096 820989.6)
9	1967	09003	09	003		Coyoacán	BJU POINT (2797862 817634.2)

¿Cuál es la distancia entre folios y estaciones de monitoreo?

```
dist <-  
  st_distance(pts_cdmx, monit) %>%  
  print()
```

El resultado es una matriz



Units: [m]								
	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]
[1,]	11916.406	15220.0657	17203.571	11168.206	6499.460	4118.933	8096.5364	27769.131
[2,]	23716.736	10120.7845	7764.506	14413.920	23056.665	20931.504	15984.9149	5166.418
[3,]	19966.993	6634.5660	4865.522	10162.473	18464.174	15957.669	10992.1981	10145.339
[4,]	20065.468	9121.5589	8624.523	10519.264	17151.901	13871.307	9408.8064	15428.959
[5,]	8115.284	12006.8501	14213.861	7620.333	3055.007	777.837	5546.9032	24897.482
[6,]	22629.547	9002.6512	6782.020	13018.107	21519.217	19156.235	14187.6347	7449.255
[7,]	20414.776	9025.2479	8330.349	10755.164	17661.391	14452.289	9888.0949	14716.066
[8,]	17925.016	6644.6396	6358.045	8200.982	15427.603	12431.889	7648.5939	14461.506
[9,]	18173.834	4653.5061	2288.633	9080.574	17772.795	16012.335	11220.4678	8491.929
[10,]	13391.319	2945.5145	4468.799	3516.512	11741.957	9556.685	4695.8396	15015.401





¿Qué nos falta?

Transformar a tibble

```
dist <- as_tibble(dist)
```

Nombres de columna

```
names(dist) <- monit$site
```

Recuperar el folio

```
bind_cols(sitios_cdmx, dist)
```



```
st_distance(pts_cdmx, monit) %>%
  as_tibble() %>%
  rename_all(list(~monit$site)) %>%
  bind_cols(pts_cdmx)
```

Simple feature collection with 143 features and 9 fields

Geometry type: POINT

Dimension: XY

Bounding box: xmin: 2782545 ymin: 804611.1 xmax: 2814894 ymax: 837532.4

Projected CRS: Mexico ITRF2008 / LCC

First 10 features:

	folio	AZC	BJU	COY	HGM	IMP	LVI	MER	TPN	geometry
1	1754	11916.406	15220.066	17203.571	11168.206	6499.460	4118.933	8096.536	27769.131	POINT (2805067 835529.4)
2	2722	23716.736	10120.784	7764.506	14413.920	23056.665	20931.504	15984.915	5166.418	POINT (2799581 812358.5)
3	8758	19966.993	6634.566	4865.522	10162.473	18464.174	15957.669	10992.198	10145.339	POINT (2801880 817228)
4	2192	20065.468	9121.559	8624.523	10519.264	17151.901	13871.307	9408.806	15428.959	POINT (2806450 820158.9)
5	3161	8115.284	12006.850	14213.861	7620.333	3055.007	777.837	5546.903	24897.482	POINT (2801141 833745)
6	8830	22629.547	9002.651	6782.020	13018.107	21519.217	19156.235	14187.635	7449.255	POINT (2801194 814034.4)
7	1938	20414.776	9025.248	8330.349	10755.164	17661.391	14452.289	9888.095	14716.066	POINT (2806145 819439)
8	8606	17925.016	6644.640	6358.045	8200.982	15427.603	12431.889	7648.594	14461.506	POINT (2804096 820989.6)
9	1967	18173.834	4653.506	2288.633	9080.574	17772.795	16012.335	11220.468	8491.929	POINT (2797862 817634.2)
10	1913	13391.319	2945.514	4468.799	3516.512	11741.957	9556.685	4695.840	15015.401	POINT (2800127 823754.8)