Escola d'Enginyeria UAB

UAB

SEMINARI 7. *Advanced Systems II*

1. OBJECTIUS

Introduir la creació de mapes amb R i concretament amb ggplot2. Els mapes són una eina de sistemes avançats de visualització. Ja vam veure a la classe de teoria alguns exemples d'ús, i avui crearem alguns mapes més.

NOTA: Si vau fer els exercicis durant la segona part de la classe teòrica del Tema 9, podeu passar directament a l'exercici 4 de l'apartat 4.

2. PAQUETS A INSTAL·LAR (per parts 3 i 4 del seminari)

Hi ha molts paquets R disponibles a CRAN (la xarxa d'arxius completa R, que és el dipòsit principal dels paquets R). Per aquest seminari, haureu d'instal·lar alguns paquets addicionals

 # Aquests són paquets que necessitareu (a més de ggplot2 i dplyr), però que probablement ja teniu. (No us molesteu a instal·lar-los si ja els teniu):

```
> install.packages (c("devtools", "stringr"))
```

#cliqueu no si us demana restart R

 # També necessitareu, alguns paquets de mapes estàndard:

```
> install.packages (c ("maps", "mapdata","ggmap"))
```

3. COM FEM MAPES

Normalment podem descompondre la creació d'un mapa en dos "problemes": Utilitzar dades d'una certa font per dibuixar el mapa (background) i afegir al mapa informació de les metadades d'una altra font.

Tractarem majorment amb classes *sp* per dates espacials (per defecte) o directament classes *sf* (simple features).

Ara que tenim els paquets instal·lats, carreguem les llibreries que no tinguem carregades:

```
> library(ggplot2)
> library(ggmap)
> library(maps)
> library(mapdata)
```

- **El paquet de mapes (maps) conté una gran quantitat de contorns de continents, països, estats i comtats que han estat en R durant molt de temps (pot ser doncs, que no estigui totalment actualitzat o que no estigui 100% correcte i això, ho haurem de tenir en compte). També conté esquemes i punts de ciutats, etc. Exemples: nz, State, world, etc.**
- **El paquet *mapdata* conté uns quants esquemes més, amb una resolució més alta.**
- **Tot i que el paquet "maps", inclou una funció de traçat, optarem per utilitzar ggplot2 per representar els mapes.**

Recordeu que ggplot2 funciona en marcs de dades (*dataframes*). Per tant, necessitarem passar les dades dels mapes a un format *dataframe* que ggplot2 entengui.

ggplot2 proporciona la funció `map_data()`.

Penseu-ho com una funció que converteix una sèrie de punts al llarg d'un esquema en un marc de dades d'aquests punts.

Sintaxi: `map_data ("nom")` on "nom" és una cadena amb el nom d'un mapa al paquet de mapes o dades de mapes

Exemple:

```
> usa <- map_data("usa")
> str(usa)
'data.frame':   7243 obs. of  6 variables:
 $ long     : num  -101 -101 -101 -101 -101 ...
 $ lat      : num  29.7 29.7 29.7 29.6 29.6 ...
 $ group    : num  1 1 1 1 1 1 1 1 1 1 ...
 $ order    : int  1 2 3 4 5 6 7 8 9 10 ...
 $ region   : chr  "main" "main" "main" "main" ...
 $ subregion: chr  NA NA NA NA ...
>
```

L'estructura d'aquests dataframes són senzilles:

- ***long* - és la longitud. A l'oest del meridià primer aquestes són negatives.**
- ***lat* – és la latitud.**
- ***order* - només mostra en quin ordre ggplot ha de "connectar els punts"**
- ***region* & subregion - indiquen quina regió o subregió envolta un conjunt de punts.**
- ***group* - és molt important. Les funcions de `ggplot2` poden adoptar un argument de grup que controla (entre altres coses) si els punts adjacents haurien d'estar connectats per línies. Si formen el mateix grup, es connecten, però si estan en grups diferents, no ho fan.**
  Essencialment, tenir punts en diferents grups significa que `ggplot` "aixeca el bolígraf" quan va entre ells.

4. POLYGON MAPS

Quan tenim la longitud i latitud dels mapes, una manera senzilla de dibuixar mapes, com veurem en l'exercici 1, és utilitzant `geom_polygon()`. Aquesta geometria ens ajuda a dibuixar contorns per diferents regions.

`geom_polygon()` dibuixa línies entre punts i "les tanca" (és a dir, dibuixa una línia des de l'últim punt fins al primer punt).

EXERCICIS

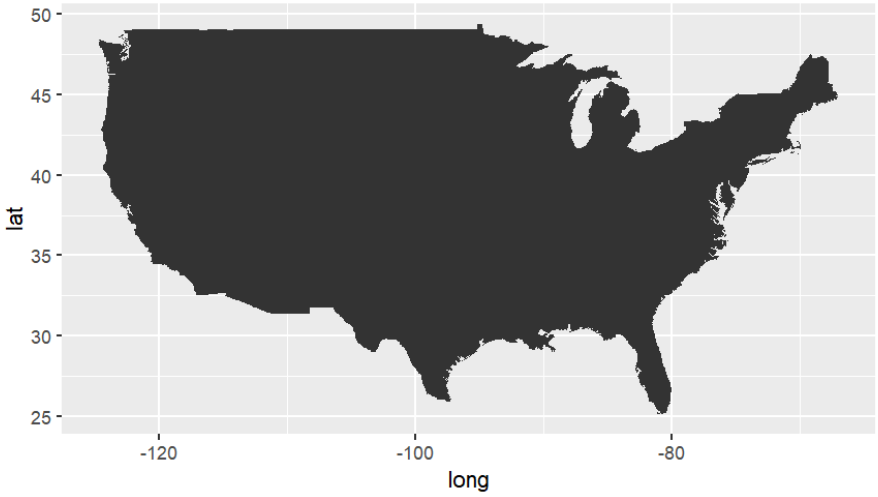
1.- Anem a testejar la comanda `geom_polygon()` per dibuixar mapes quan tenim dades espacials que fan ús de la longitud-latitud. Per fer ús de ggplot, recordeu que necessitem les dades, el mapeig (al nostre sistema de coordenades com a mínim) i la geometria. En aquest cas, el nostre sistema de coordenades és `x =`

long **i** `y = lat`**. Per tant, com sempre, heu de fer un mapeig (usant l'estètica aes) per representar les dades en el vostre sistema de coordenades. A més, fent ús de *group* en el vostre mapeig, agrupeu les dades espacials de manera que es connectin els punts que han d'estar connectats per línies. Finalment, si agregueu l'argument *coord_quickmap()*us permet ajustar els eixos per assegurar-nos que la longitud i la latitud es representen a la mateixa escala.**

(a) Seguint les indicacions que se us ha donat a l'enunciat, dibuixeu el mapa de USA

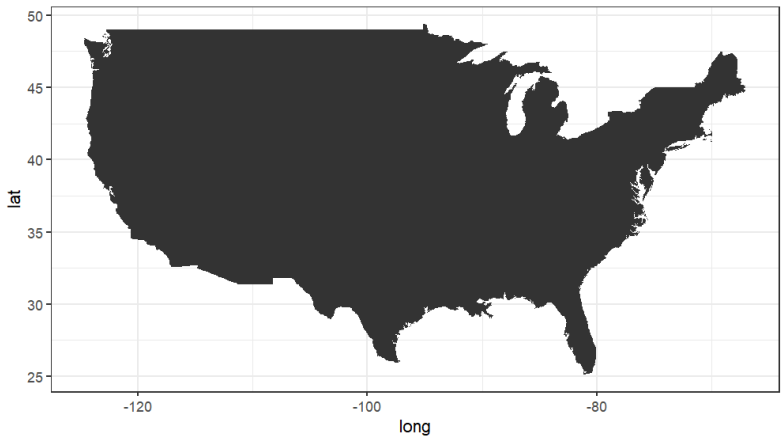
NOTA: Recordeu assignar primer a la variable de nom *usa*, un marc de dades (*dataframe*) de USA. Això podeu fer-ho gràcies a la funció `map_data` que heu vist més amunt:

```
> usa <- map_data("usa")
>ggplot(usa)+ aes(x=long, y = lat, group = group)+geom_polygon()+coord_quickmap()
```



(b) Utilitzeu el clàssic layer de llum fosca per a `ggplot2` adequat per als mapes: `theme_set(theme_bw())`

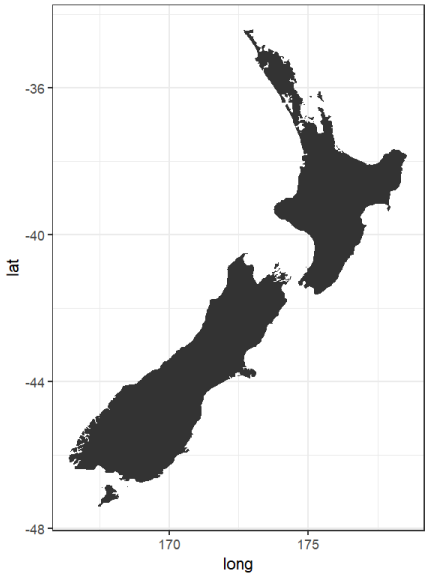
```
> ggplot(usa)+ aes(x=long, y = lat, group = group)+geom_polygon()+coord_quickmap()+theme_set(theme_bw())
```



(c) Dibuixeu el mapa de Nova Zelanda (*nz*) anàlogament

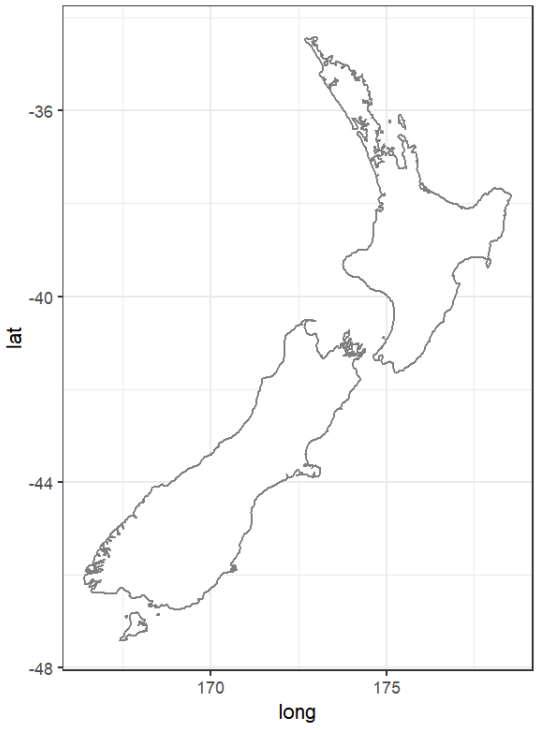
> nz<- **map_data("nz")**

> ggplot (**nz**) +aes(x=long,y=lat,group=group) + geom_polygon() + **coord_quickmap() + theme_set(theme_bw())**



(d) Ara en canvi feu us de *`fill`* i *`colour`* que ja hem usat en altres geometries per omplir de blanc ("`white`") l'interior del mapa i posar els contorns en gris ("`grey50`")

> ggplot (nz) +aes(x=long,y=lat,group=group) +geom_polygon**(fill="white", colour="grey50")**+coord_quickmap()



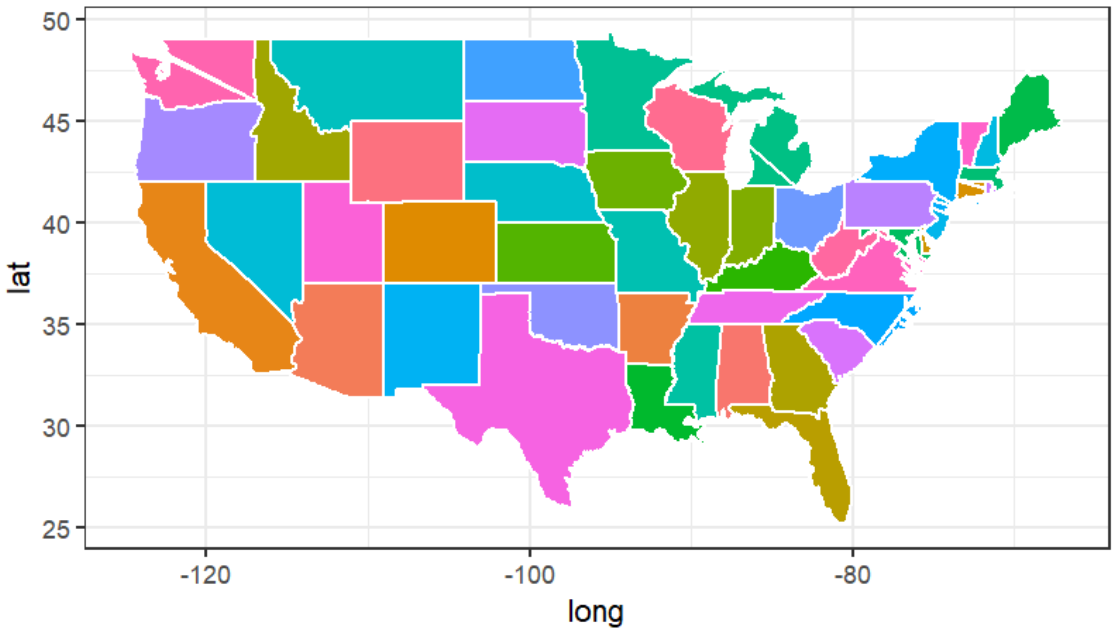
2.- Fent ús de la funció, map_data, també podem obtenir un dataframe que ens indiqui les fronteres estatals del mapa 'usa':

```
>  states <- map_data("state")
```

```
~/
> head(states)
       long      lat group order  region subregion
1 -87.46201 30.38968     1     1 alabama      <NA>
2 -87.48493 30.37249     1     2 alabama      <NA>
3 -87.52503 30.37249     1     3 alabama      <NA>
4 -87.53076 30.33239     1     4 alabama      <NA>
5 -87.57087 30.32665     1     5 alabama      <NA>
6 -87.58806 30.32665     1     6 alabama      <NA>
>
```

Pinteu de colors els estats (atribut=*region*) i poseu les fronteres en blanc. Afegiu `guides(fill=FALSE)` **per tal d'eliminar la llegenda.**
NOTA: En algunes versions de R surt un *warning*. Si és el cas, useu:
`guides(fill= "none")`

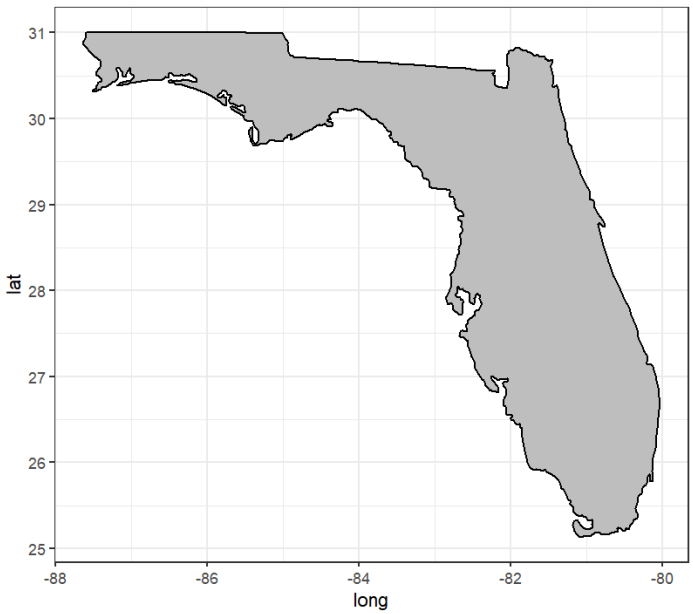
```
>  ggplot(data = states) + geom_polygon(aes(x = long, y = lat, fill =
region), color = "white")+coord_quickmap()+guides(fill=FALSE)
> ggplot(data = states) + geom_polygon(aes(x = long, y = lat, fill =
region), color = "white")+coord_quickmap()+guides(fill="none")
```



3.- També podem crear un subconjunt amb la informació d' un estat (atribut='region') que ens interessi, fent servir:
`fo_df <- subset(states, region == "florida") # estat de Florida`

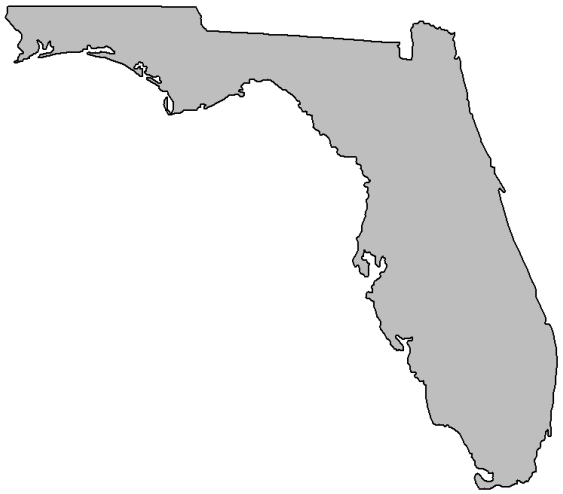
(a) Dibuixeu el mapa de l'estat de Florida, pinteu l'interior en gris i el contorn en negre.

```
>  ggplot(data = fo_df, mapping = aes(x = long, y = lat)) +
coord_quickmap()+ geom_polygon(color = "black", fill = "gray")
```



(b) Afegiu l'argument `theme_nothing()`. Què us fa?

```
> ggplot(data = fo_df, mapping = aes(x = long, y = lat)) +
coord_quickmap()+ geom_polygon(color = "black", fill =
"gray")+theme_nothing()
```



Treu els eixos i graella

(c) Repetiu el procés amb California, i assigneu a una variable de nom 'CA_base' un ggplot d'un mapa com el de l'apartat anterior però de California (amb background buit). NOTA: Primer extraieu la informació de l'estat en un subconjunt com s'ha fet abans

Nota 2: Els noms de les regions van en minúscules

```
> CA_df <- subset(states, region == "california")
```

```
> CA_base <- ggplot(data = CA_df, mapping = aes(x = long, y = lat,
group = group)) + coord_quickmap()+ geom_polygon(color = "black", fill
= "gray")+ theme_nothing()
> CA_base
```

(d) Extraieu ara en un subconjunt els comptats de California (`"county"`):

```
> counties <- map_data("county")
> CA_county <- subset(counties, region == "california")
```

```
> head(CA_county)
          long      lat group order     region subregion
6965 -121.4785 37.48290   157  6965 california   alameda
6966 -121.5129 37.48290   157  6966 california   alameda
6967 -121.8853 37.48290   157  6967 california   alameda
6968 -121.8968 37.46571   157  6968 california   alameda
6969 -121.9254 37.45998   157  6969 california   alameda
6970 -121.9483 37.47717   157  6970 california   alameda
>
```

(e) Afegiu a la vostra variable CA_base creada en l'apartat anterior un `geom_polygon()` que contingui les dades del subconjunt CA_county, i el color de les fronteres entre comptats estigui blanc (deixeu `fill=NA`).

```
> CA_base + geom_polygon(data = CA_county, fill = NA, color = "white")
```

Podeu tornar a situar la frontera estatal, fent:

```
> CA_base + geom_polygon(data = CA_county, fill = NA, colour =
"white") + geom_polygon(fill = NA, colour = "black")
```

Podeu trobar altres paquets de R i llibreries per fer mapes i l'explicació del seu us a: Chapter 9 Making maps with R | Geocomputation with R (geocompx.org), **i a:** https://www.earthdatascience.org/courses/earth-analytics/spatial-data-r/make-maps-with-ggplot-in-R/

O aprofundir més seguint el material en que s'ha basat aquesta part de la classe:

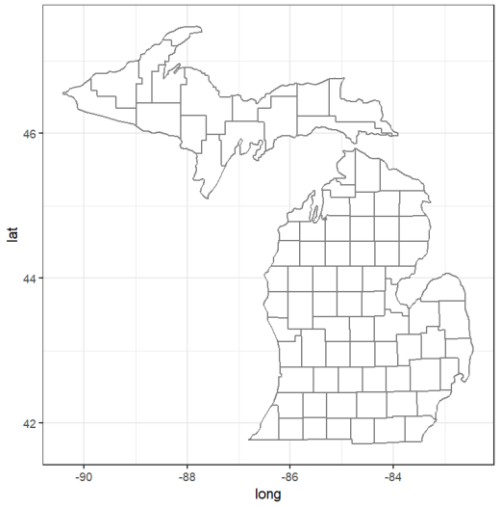
Making Maps With R · Reproducible Research. (eriqande.github.io)

3.- En aquest exercici carreguem un mapa de Michigan que porta incorporats els contorns dels comptats de Michigan

```
mi_counties <- map_data("county", "michigan")
```

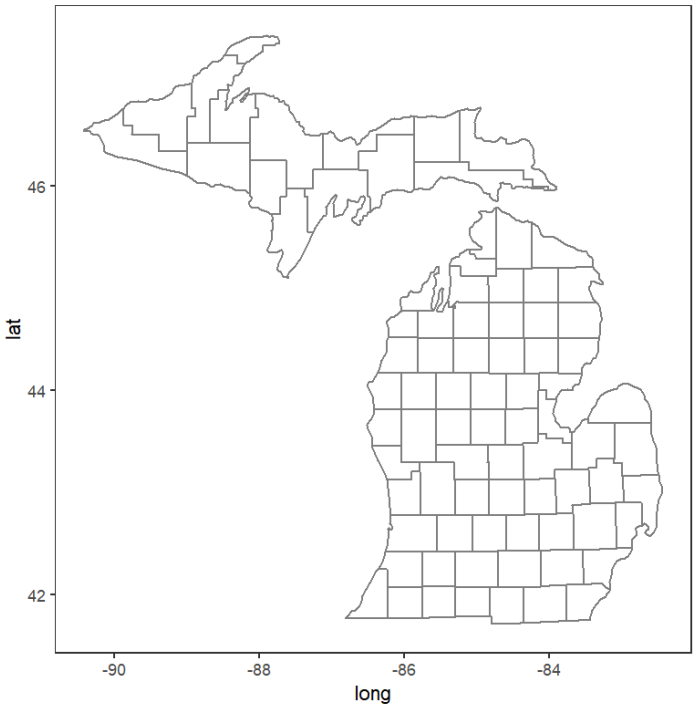
a) Feu us de `geom_polygon()` per dibuixar el mapa. Poseu el fons del mapa blanc i el contorn en gris (`"grey50"`).

```
> mich<- ggplot(mi_counties)+aes(long, lat, group = group) +geom_polygon(fill = "white", colour = "grey50") + coord_quickmap()
> mich
```



b) Afegiu `theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank())`

```
> mich+theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank())
```



5. PAQUET "SIMPLE FEATURES MAPS" (SF)

Hi ha algunes limitacions amb l'enfocament descrit en els apartats 3 i 4 anteriors. La principal limitació és el fet que el format de dades simple "longitud-latitud" no

s'acostumen a utilitzar en els mapes del món real. Les dades vectorials per als mapes es codifiquen normalment mitjançant l'estàndard de "funcions simples" (sf, de les sigles en anglès) produït per l'Open Geospatial Consortium.

El paquet `sf` desenvolupat per Edzer Pebesma (https://github.com/r-spatial/sf) ofereix un conjunt d'eines excel·lent per treballar amb aquestes dades, i les funcions `geom_sf()` i `coord_sf()` de ggplot2 estan dissenyades per treballar conjuntament amb el paquet `sf`.

Per a trames simples, només necessitareu `geom_sf()` (una geometria inusual que dibuixarà diferents objectes geomètrics en funció de les sf que hi hagi a les dades: podeu obtenir punts, línies o polígons). Per afegir text i etiquetes, podeu utilitzar `geom_sf_text()` i `geom_sf_label()`, respectivament.

Hi ha tres maneres de proporcionar l'estètica de la geometria amb `geom_sf()` :

1. **No feu res: per defecte `geom_sf()` assumeix que s'emmagatzema a la columna de geometria.**
2. **Passeu explícitament un objecte `sf` a l'argument de dades. Això utilitzarà la columna de geometria primària, sense importar com es digui.**
3. **Subministreu-ne una de vostra amb `aes(geometry = my_column)`**

Si no s'especifica el sistema de referència de coordenades (CRS) en el qual s'han de projectar totes les dades abans de representar-les, utilitzarà el CRS definit a la primera capa `sf` de la trama.

`coord_sf()` garanteix que totes les capes utilitzen un CRS comú. Podeu especificar-lo mitjançant el paràmetre `crs`, o bé `coord_sf()`. Ho agafarà de la primera capa que defineix un CRS.

Per introduir aquestes funcions, anem a utilitzar el paquet `ozmaps` de Michael Summer (https://github.com/mdsumner/ozmaps/). Són dades que es poden dibuixar o accedir directament com a objectes de característiques simples. El paquet inclou funcions senzilles per a mapes de països o estats d'Austràlia i conjunts de dades de regions administratives de l'Oficina d'Estadística d'Austràlia <https://www.abs.gov.au/>. Les capes inclouen les divisions electorals i àrees de govern local, etc. Tot i que totes elles estan simplificades a partir de les fonts originals, contenen suficients detall per permetre la cartografia d'un municipi local.

Per utilitzar el paquet ozmaps anem a carregar-lo. Carregarem també el paquet sf i cridarem ambdues llibreries, `ozmaps` i `sf`:

```
> install.packages("ozmaps")
```

 # la versió actualitzada de *github*

```
> devtools::install_github("mdsumner/ozmaps")
```
#cliqueu 1 quan us demani

 # Ignoreu els warnings de carregar llibreries.

```
> install.packages("sf")
```

Ara que tenim els paquets instal·lats, carreguem les llibreries:

```
> library(ozmaps)
> library(sf)
```

Assigneu:

```
> oz_states <- ozmaps::ozmap_states
```

Verifiqueu que teniu la *sf*:

```
> oz_states
Simple feature collection with 9 features and 1 field
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: 105.5507 ymin: -43.63203 xmax: 167.9969 ymax: -9.229287
Geodetic CRS:  GDA94
# A tibble: 9 x 2
  NAME                                                        geometry
* <chr>                                                  <MULTIPOLYGON [°]>
1 New South Wales              (((150.7016 -35.12286, 150.6611 -35.117~
2 Victoria                     (((146.6196 -38.70196, 146.6721 -38.702~
3 Queensland                   (((148.8473 -20.3457, 148.8722 -20.3757~
4 South Australia              (((137.3481 -34.48242, 137.3749 -34.468~
5 Western Australia            (((126.3868 -14.01168, 126.3625 -13.982~
6 Tasmania                     (((147.8397 -40.29844, 147.8902 -40.302~
7 Northern Territory           (((136.3669 -13.84237, 136.3339 -13.839~
8 Australian Capital Territory (((149.2317 -35.222, 149.2346 -35.24047~
9 Other Territories            (((167.9333 -29.05421, 167.9188 -29.034~
>
```

1.- Donades les dades en aquest format, podem utilitzar `geom_sf()` i `coord_sf()` per dibuixar un mapa útil sense especificar cap paràmetre o fins i tot, sense necessitat de declarar explícitament cap estètica/mapeig.

(a) Dibuixeu el mapa de oz_states tot traient la graella del fons. Cal fer ús de `coord_sf()`? Si la resposta és negativa argumenta per què?

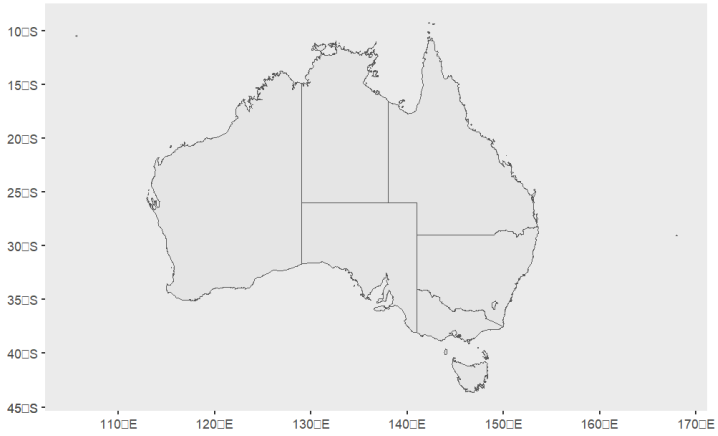
Si ens fixem oz_states utilitza el sistema de coordenades CRS

```
> oz_states
Simple feature collection with 9 features and 1 field
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: 105.5507 ymin: -43.63203 xmax: 167.9969 ymax: -9.229287
Geodetic CRS:  GDA94
# A tibble: 9 x 2
  NAME                                                        geometry
* <chr>                                                  <MULTIPOLYGON [°]>
1 New South Wales              (((150.7016 -35.12286, 150.6611 -35.117~
2 Victoria                     (((146.6196 -38.70196, 146.6721 -38.702~
3 Queensland                   (((148.8473 -20.3457, 148.8722 -20.3757~
4 South Australia              (((137.3481 -34.48242, 137.3749 -34.468~
5 Western Australia            (((126.3868 -14.01168, 126.3625 -13.982~
6 Tasmania                     (((147.8397 -40.29844, 147.8902 -40.302~
7 Northern Territory           (((136.3669 -13.84237, 136.3339 -13.839~
8 Australian Capital Territory (((149.2317 -35.222, 149.2346 -35.24047~
9 Other Territories            (((167.9333 -29.05421, 167.9188 -29.034~
>
```

Com hem dit abans: “Si no s'especifica el sistema de referència de coordenades (CRS) en el qual s'han de projectar totes les dades abans de representar-les, utilitzarà el CRS definit a la primera capa `sf` de la trama.” Per tant, afegir o no el `coord_sf` no varia, en aquest cas, la visualització :

```
>      ggplot(oz_states)+geom_sf()+theme(panel.grid.major      =
element_blank(), panel.grid.minor = element_blank())
```

```
>   ggplot(oz_states)+geom_sf()+coord_sf()+theme(panel.grid.major
= element_blank(), panel.grid.minor = element_blank())
```



(b) En segon lloc, extraieu els límits electorals d'una forma simplificada utilitzant la funció `ms_simplify()` del paquet `rmapshaper`. Nota: Instal·leu primer el paquet `rmapshaper`. Aquí teniu més informació d'aquest paquet: https://cran.r-project.org/web/packages/rmapshaper/rmapshaper.pdf

La funció ms_simplify ens permet simplificar polígons o línies.

Per fer aquest apartat, us avancem la resposta, simplement heu de fer: `oz_votes <- rmapshaper::ms_simplify(ozmaps::abs_ced)`

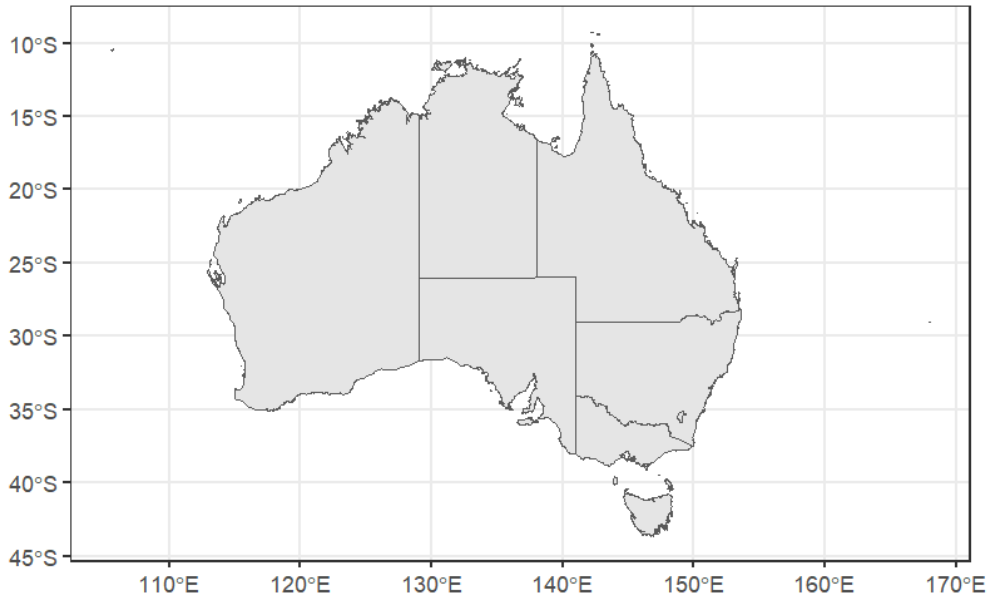
Compareu `ozmaps::abs_ced` **i** `oz_votes`**. Què ha simplificat** `ms_simplify`**?**

```
> ozmaps::abs_ced
Simple feature collection with 151 features and 1 field
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: 96.81703 ymin: -43.65856 xmax: 167.9969 ymax: -9.219937
Geodetic CRS:  GDA94
First 10 features:
        NAME                    geometry
1      Banks MULTIPOLYGON (((151.0156 -3...
2     Barton MULTIPOLYGON (((151.1681 -3...
3  Bennelong MULTIPOLYGON (((151.0511 -3...
4    Berowra MULTIPOLYGON (((151.1786 -3...
5   Blaxland MULTIPOLYGON (((151.0301 -3...
6  Bradfield MULTIPOLYGON (((151.1001 -3...
7     Calare MULTIPOLYGON (((149.1198 -3...
8    Chifley MULTIPOLYGON (((150.8666 -3...
9       Cook MULTIPOLYGON (((151.1548 -3...
10    Cowper MULTIPOLYGON (((153.1503 -3...
> oz_votes
Simple feature collection with 151 features and 1 field
Geometry type: GEOMETRY
Dimension:     XY
Bounding box:  xmin: 105.5507 ymin: -43.63203 xmax: 153.6299 ymax: -9.229287
Geodetic CRS:  GDA94
First 10 features:
        NAME                    geometry
1      Banks POLYGON ((151.0156 -33.9820...
2     Barton POLYGON ((151.1681 -33.9201...
3  Bennelong POLYGON ((151.0511 -33.8242...
4    Berowra POLYGON ((151.1597 -33.6605...
5   Blaxland POLYGON ((151.0301 -33.8351...
6  Bradfield POLYGON ((151.1001 -33.7535...
7     Calare POLYGON ((149.1198 -33.8878...
8    Chifley POLYGON ((150.8666 -33.6525...
9       Cook POLYGON ((151.1548 -33.9666...
10    Cowper POLYGON ((153.1503 -30.2398...
>
```

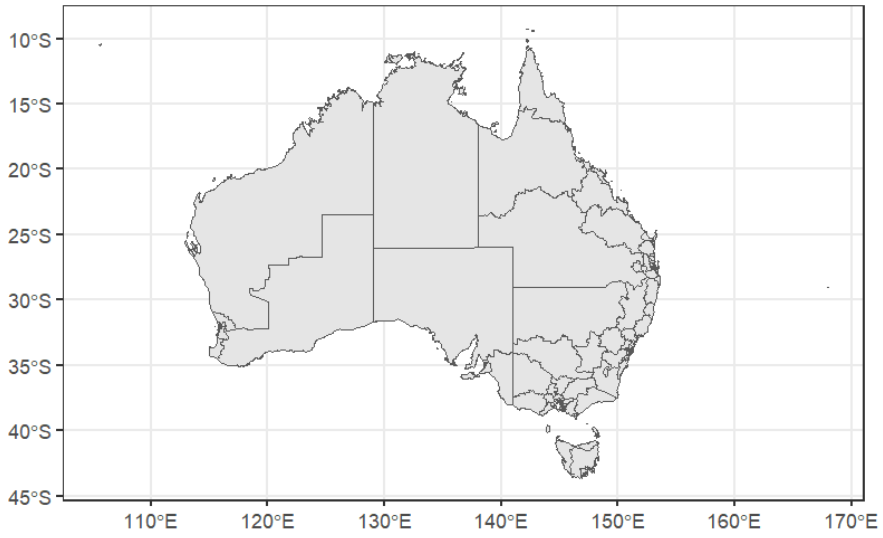
Ens ha simplificat clarament els polígons.

(c) Ara que teniu ambdós conjunts de dades per representar les fronteres estatals (oz_states) i electorals (oz_votes) en un mateix mapa, afegiu dues capes `geom_sf()`, una per les dades contingudes en oz_states i l'altra per les dades contingudes en oz_votes

```
> ggplot() + geom_sf(data = oz_states)
```



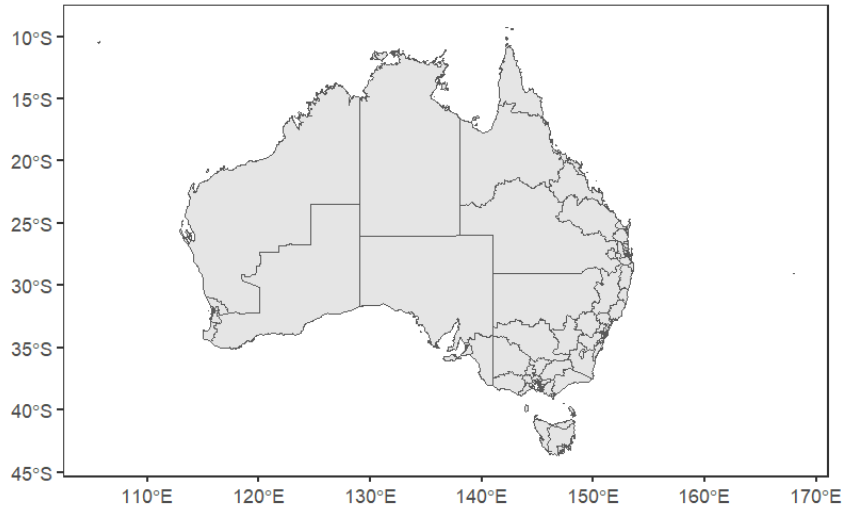
```
> vots<- ggplot() + geom_sf(data = oz_states) +geom_sf(data = oz_votes)
```



Nota: Fixeu-vos no cal `coord_sf()`

```
> vots+coord_sf()
```

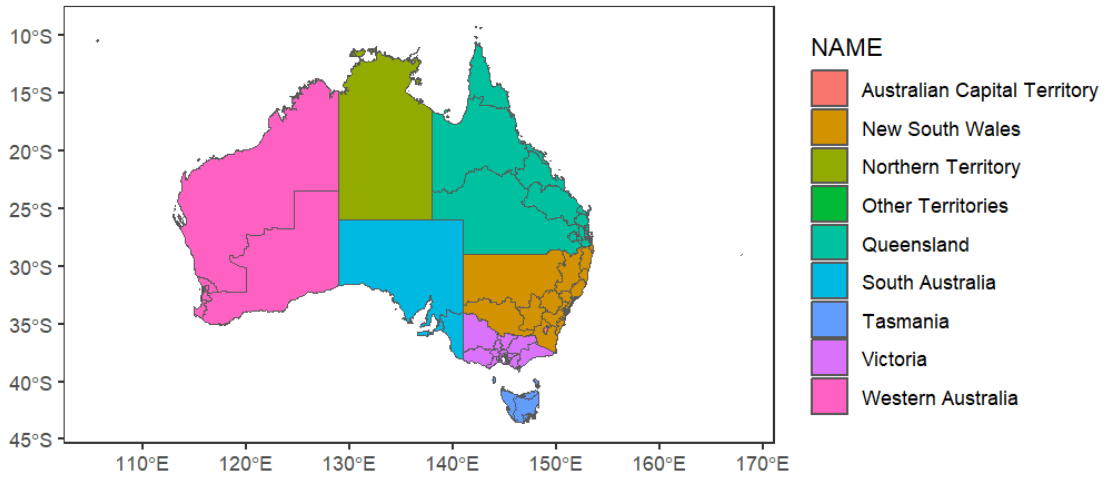
```
> vots + theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank())
```



(d) Afegiu l'estètica necessària per pintar l'interior (`fill`) dels estats segons el seu nom (NAME).

NOTA: Per evitar el conflicte amb la informació de `oz_votes`, poseu en la capa `geom_sf()` de `oz_votes`: `geom_sf(data = oz_votes, fill=NA)`.

```
> ggplot() + geom_sf(data = oz_states, aes(fill=NAME))
+geom_sf(data = oz_votes, fill=NA) +
coord_sf()+theme(panel.grid.major = element_blank(),
panel.grid.minor = element_blank())
```



IMPORTANT: En aquest cas, la variable NAME és una variable categòrica i no transmet cap informació addicional, però el mateix enfocament es podria utilitzar per visualitzar altres tipus de metadates d'àrea. Per exemple, si oz_states tingués una columna addicional que especifiques el nivell de persones empleades o en atur a cada estat, podríem assignar l'estètica de de color a aquesta variable.

2.- En aquest exercici anem a dibuixar un mapa electoral de Sydney partint del mapa que tenim. Per això hem de fer tres passos:

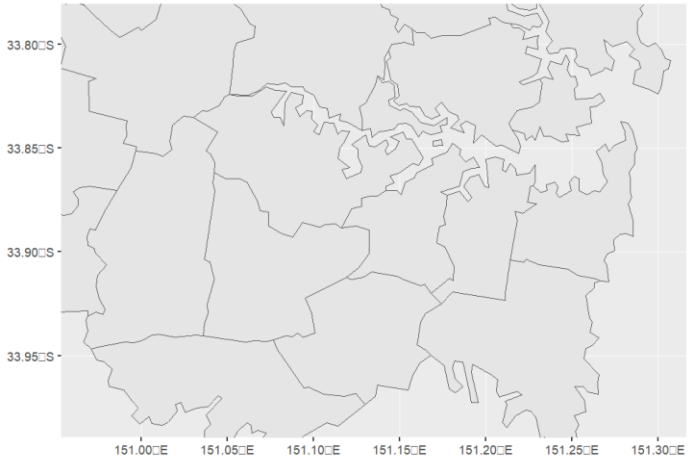
(a) Primer, hem d'extreure les dades del mapa pels electors rellevants. Copieu el següent filtratge:

```
> sydney_map <- ozmaps::abs_ced %>% filter(NAME %in% c("Sydney", "Wentworth", "Warringah", "Kingsford Smith", "Grayndler", "Lowe", "North Sydney", "Barton", "Bradfield", "Banks", "Blaxland", "Reid", "Watson", "Fowler", "Werriwa", "Prospect", "Parramatta", "Bennelong", "Mackellar", "Greenway", "Mitchell", "Chifley", "McMahon"))
```

NOTA: Recordeu carregar la llibreria necessària per `filter` si no ho heu fet

(b) Després hem de buscar la regió que ens interessa, Sydney. Per això, simplement feu zoom a la regió de Sydney especificant `xlim = c(150.97, 151.3), ylim = c(-33.98, -33.79)` a `coord_sf()`

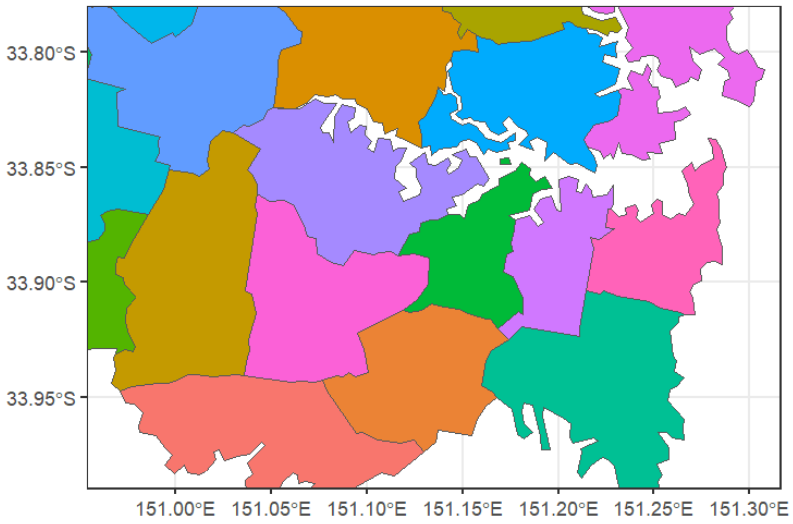
```
> ggplot(sydney_map) +geom_sf() + coord_sf(xlim = c(150.97, 151.3), ylim = c(-33.98, -33.79))
```



(c) Tot utilitzant la regió de Sydney que heu creat en l'apartat (b), pinteu cada subregió segons el seu NAME, poseu dins del `geom_sf` l'argument `show.legend = FALSE`, per amagar la llegenda

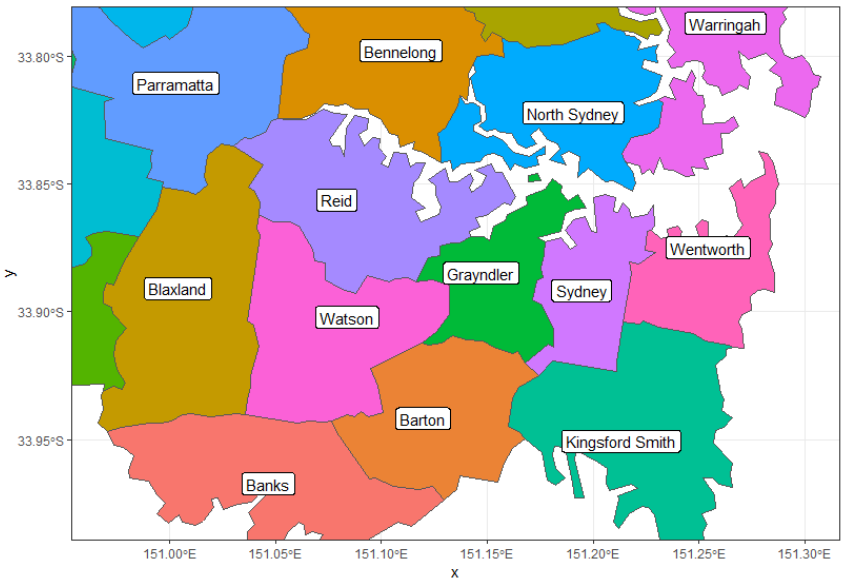
```
> syd <- ggplot(sydney_map) +geom_sf(aes(fill = NAME), show.legend = FALSE) + coord_sf(xlim = c(150.97, 151.3), ylim = c(-33.98, -33.79))
```

```
> syd
```



(d) Finalment, utilitzeu `geom_sf_label()` per superposar les etiquetes amb el nom de cada electorat (poseu l'estètica `label=NAME`)

```
> syd +geom_sf_label(aes(label = NAME))
```



3.- Ens podria interessar per exemple mostrar les ubicacions de les capitals australianes al mapa anterior.

Suposem que tenim les dades de les ciutats junt a la seva latitud i longitud. Afegiu una altra geometria, per exemple, `geom_point()` que mostri aquesta informació.

(a) Construïu el següent dataframe fent una `tibble`, i assigneu-lo a una variable de nom oz_capitals. Se us dona el codi :

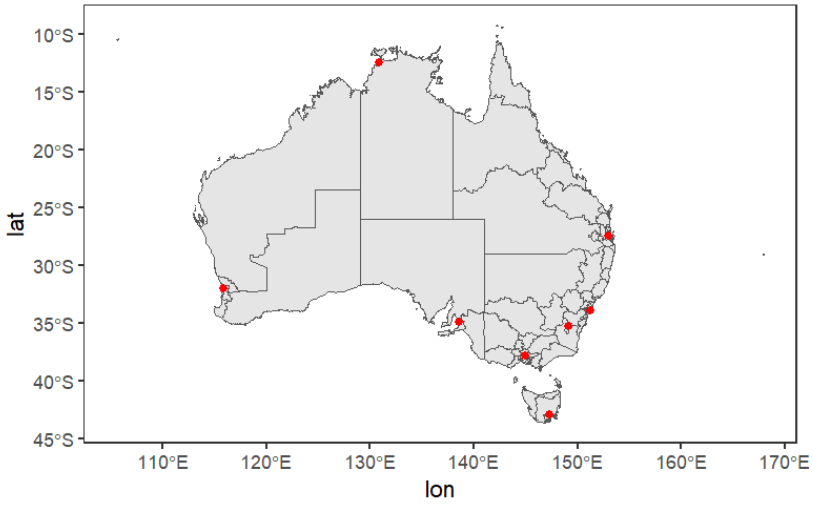
```
> oz_capitals <- tibble::tribble(
  ~city,  ~lat,     ~lon,
  "Sydney",  -33.8688, 151.2093,
  "Melbourne",-37.8136, 144.9631,
  "Brisbane", -27.4698, 153.0251,
  "Adelaide", -34.9285, 138.6007,
  "Perth",    -31.9505, 115.8605,
  "Hobart",   -42.8821, 147.3272,
  "Canberra", -35.2809, 149.1300,
  "Darwin",   -12.4634, 130.8456,
)
```

```
> oz_capitals
# A tibble: 8 x 3
  city        lat   lon
  <chr>     <db1> <db1>
1 Sydney    -33.9  151.
2 Melbourne -37.8  145.
3 Brisbane  -27.5  153.
4 Adelaide  -34.9  139.
5 Perth     -32.0  116.
6 Hobart    -42.9  147.
7 Canberra  -35.3  149.
8 Darwin    -12.5  131.
>
```

(b) Afegiu la informació del dataframe amb punts de color vermell al mapa d'Australia que contenia les fronteres estatals (oz_states) i electorals (oz_votes)

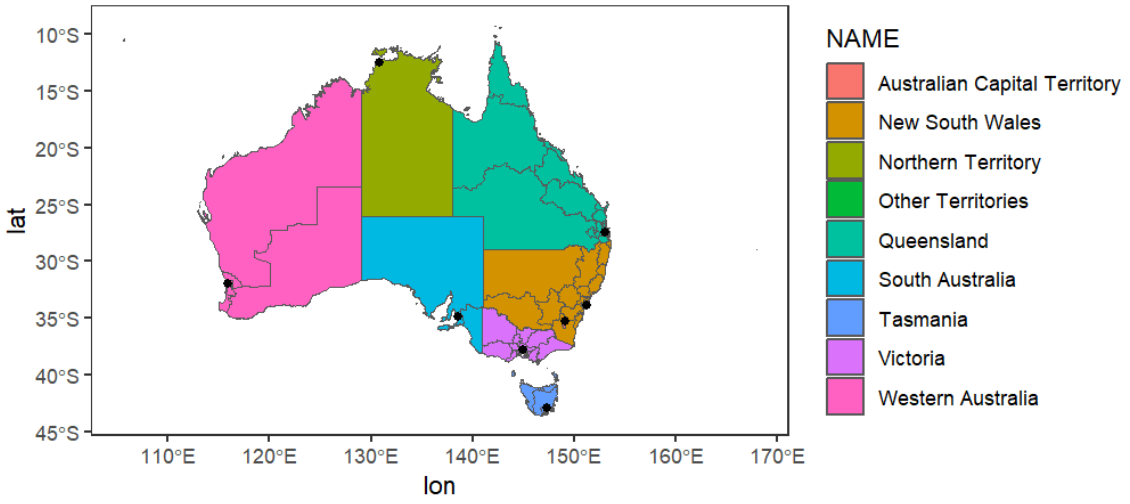
Seguint el que hem fet en l'exercici 1 d'aquesta secció i afegint `geom_point()`

```
> ggplot() + geom_sf(data = oz_states) +geom_sf(data = oz_votes) + geom_point(data = oz_capitals, mapping = aes(x = lon, y = lat), colour = "red")+coord_sf()+theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank())
```



O inclús:

```
>ggplot() + geom_sf(data = oz_states, aes(fill=NAME))
+geom_sf(data = oz_votes, fill=NA) + geom_point(data =
oz_capitals, mapping = aes(x = lon, y = lat), colour =
"black")+coord_sf()+theme(panel.grid.major = element_blank(),
panel.grid.minor = element_blank())
```



4.- Per aquest exercici anem a fer servir un arxiu georeferenciat amb les comunes de la Ciutat Autònoma de Buenos Aires. L'arxiu està disponible online en format *geojson*, un format estàndard de representació de dades geogràfiques que és fàcil d'usar

```
> Barris <-
st_read('https://bitsandbricks.github.io/data/CABA_barrios.geojson')
```

```
> head (Barris)
Simple feature collection with 6 features and 4 fields
Geometry type: POLYGON
Dimension:     XY
Bounding box:  xmin: -58.50617 ymin: -34.63064 xmax: -58.41192 ymax: -34.57829
Geodetic CRS:  WGS 84
            BARRIO COMUNA PERIMETRO    AREA                       geometry
1        CHACARITA     15  7725.695 3118101 POLYGON ((-58.45282 -34.595...
2         PATERNAL     15  7087.513 2229829 POLYGON ((-58.46558 -34.596...
3     VILLA CRESPO     15  8132.699 3613584 POLYGON ((-58.42375 -34.597...
4 VILLA DEL PARQUE     11  7705.390 3399596 POLYGON ((-58.49461 -34.614...
5          ALMAGRO      5  8537.901 4050752 POLYGON ((-58.41287 -34.614...
6        CABALLITO      6 10990.964 6851029 POLYGON ((-58.43061 -34.607...
>
```

Familiaritzeu-vos amb les variables

El dataframe té 48 files i 5 columnes. Una fila per barri i una columna per cada variable disponible (Comuna- unitat administrativa, perímetre i àrea que ens informen sobre les dimensions del polígon cobert per cada barri, i 'geometry' de tipus espacial i que conté les dades amb les seves coordenades geogràfiques)

Carreguem les dades de la població en cada barri: `poblacio <- read.csv("https://bitsandbricks.github.io/data/caba_pob_barrios_2010.csv")`

Familiaritzeu-vos també amb les variables

```
> head(poblacio)
     BARRIO POBLACION
1 AGRONOMIA      13912
2   ALMAGRO     131699
3 BALVANERA     138926
4  BARRACAS      89452
5  BELGRANO     126267
6      BOCA      45113
>
```

tenim el nom de cada barri i la seva població.

D'altra banda, anem a carregar dades provinents dels registres del Sistema Únic d'Atenció Ciutadana (o SUACI), una plataforma del Govern de la Ciutat que administra els contactes iniciats per ciutadans. Al portal de dades obertes de la Ciutat es publica cada un dels contactes rebuts per SUACI any a any.

```
> suaci2018 <- read.csv('https://bitsandbricks.github.io/data/gcba_suaci_2018.csv')
```

```
> head(suaci2018)
     BARRIO CONTACTOS
1 AGRONOMIA      7378
2   ALMAGRO     31420
3 BALVANERA     28616
4  BARRACAS     23106
5  BELGRANO     46936
6      BOCA     11495
>
```

Familiaritzeu-vos amb les noves variables altre cop

Veiem que la taula té 48 files (una per cada barri de la ciutat) i 2 columnes (una amb el nom de barri, i una altra amb la quantitat total de contactes registrats el 2018).

Com el nostre dataframe de contactes a SUACI i poblacio tenen una columna amb el mateix nom que conté les mateixes categories, sumar la informació de població a suaci2018 és tan fàcil com usar la funció de *dplyr* `left_join`: `new <- left_join(suaci2018, poblacio)`

```
> head(new)
     BARRIO CONTACTOS POBLACION
1 AGRONOMIA       7378      13912
2   ALMAGRO      31420     131699
3 BALVANERA      28616     138926
4  BARRACAS      23106      89452
5  BELGRANO      46936     126267
6      BOCA      11495      45113
>
```

(a) Sumeu la informació de Barris per a que us quedi:

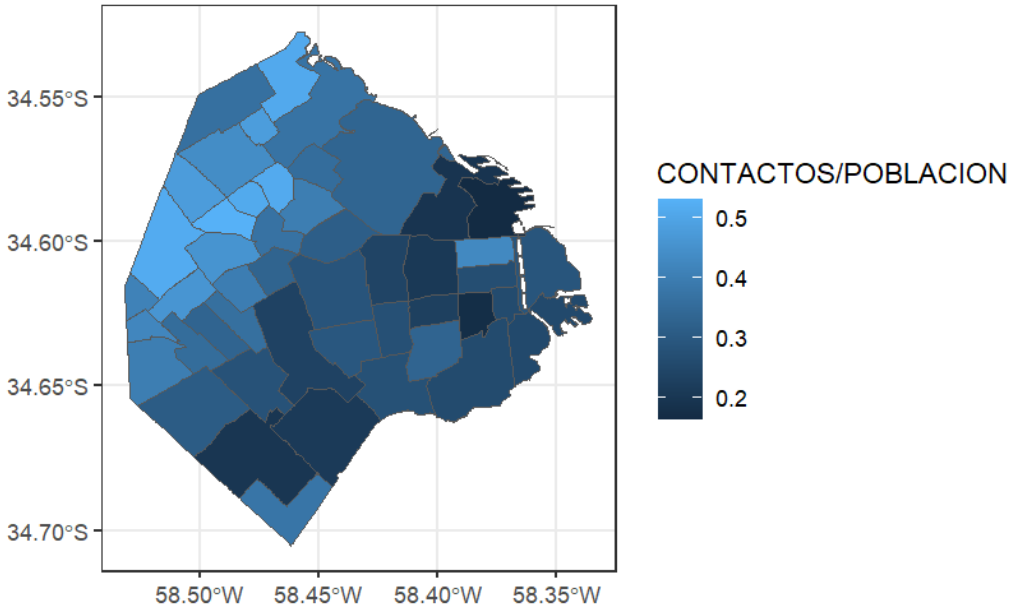
```
Simple feature collection with 6 features and 6 fields
Geometry type: POLYGON
Dimension:     XY
Bounding box:  xmin: -58.50617 ymin: -34.63064 xmax: -58.41192 ymax: -34.57829
Geodetic CRS:  WGS 84
            BARRIO COMUNA PERIMETRO    AREA CONTACTOS POBLACION
1        CHACARITA     15  7725.695 3118101     11076     27761
2         PATERNAL     15  7087.513 2229829      7255     19717
3     VILLA CRESPO     15  8132.699 3613584     25748     81959
4 VILLA DEL PARQUE     11  7705.390 3399596     25224     55273
5          ALMAGRO      5  8537.901 4050752     31420    131699
6        CABALLITO      6 10990.964 6851029     50301    176076
                       geometry
1 POLYGON ((-58.45282 -34.595...
2 POLYGON ((-58.46558 -34.596...
3 POLYGON ((-58.42375 -34.597...
4 POLYGON ((-58.49461 -34.614...
5 POLYGON ((-58.41287 -34.614...
6 POLYGON ((-58.43061 -34.607...
>
```

La comanda seria molt semblant a l'anterior:

```
> new <- left_join(Barris, new)
```

(b) Feu una comanda com la que heu fet abans per dibuixar el mapa amb `geom_sf`. Però ara les dades són el vostre nou dataframe creat en l'apartat (a). En aquest apartat se us demana dibuixar un mapa amb els barris de la ciutat que mostri la quantitat de sol·licituds de la ciutadania per càpita. És a dir que el gràfic estigui pintat segons la ratio entre contactes i població (contactes/població). PISTA: Feu un mapeig pintant el gràfic segons la relació desitjada

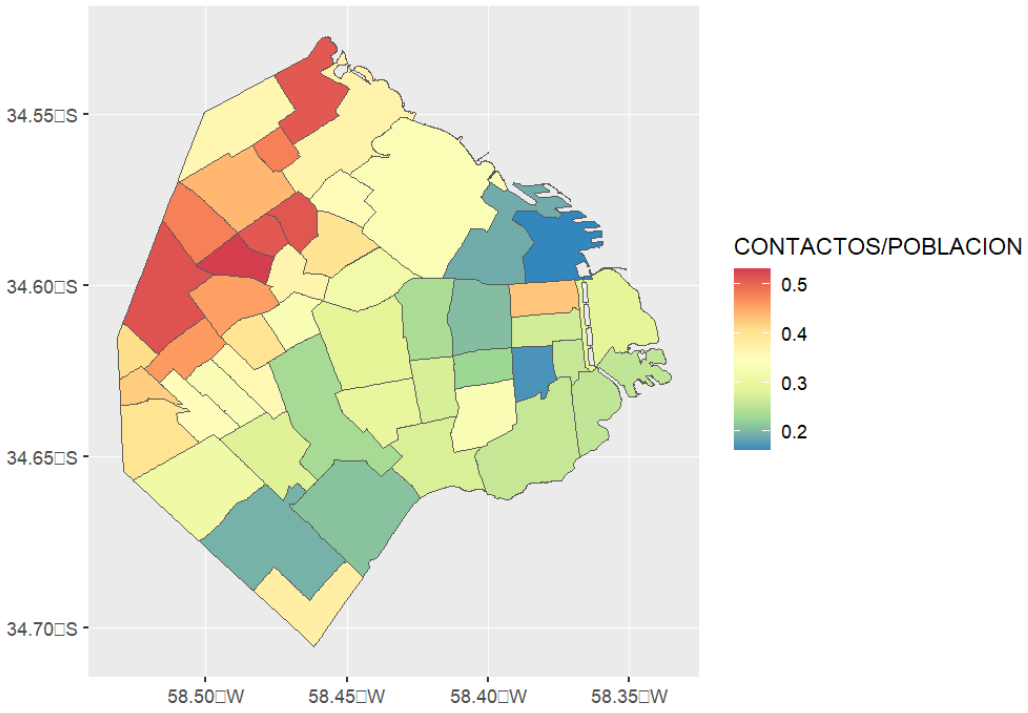
```
> ggplot(new) +geom_sf(aes(fill = CONTACTOS / POBLACION))+coord_sf()
```



(c) Finalment, definirem la paleta de colors a utilitzar en l'estètica fill. Per exemple, podeu triar una escala anomenada "Spectral", que va de el blau al vermell i és molt usada quan es vol ressaltar la divergència d'una variable (és una possibilitat però heu vist més sobre el bon us dels colors a la classe de Colors d'en Guillermo).

Per fer aquest canvi afegiu a la vostra comanda de l'exercici (b) l'escala amb: `scale_****_distiller(palette = "Spectral").` **Trieu adequadament** `****`

```
> ggplot(new) + geom_sf(aes(fill = CONTACTOS / POBLACION))+coord_sf()+scale_fill_distiller(palette = "Spectral")
```



6. EXTRA: (MINI) Introducció a mapes interactius

Mapview proporciona funcions per crear de manera molt ràpida i convenient visualitzacions interactives de dades espacials. El seu objectiu principal és omplir el buit de representació interactiva ràpida (no de qualitat de presentació) per examinar i investigar visualment tant els aspectes de les dades espacials, les geometries com els seus atributs. També es pot considerar una API dirigida per dades per al paquet `leaflet` ja que renderitzarà automàticament els tipus de mapes correctes, segons el tipus de dades (punts, línies, polígons). A més, fa ús de funcionalitats de representació avançades que permeten visualitzar dades molt més grans del que és possible amb `leaflet`.

Finalment, podeu usar `st_as_sf` per convertir un objecte que vosaltres creeu a un objecte sf. I `st_crs`, per recuperar el sistema de referència de coordenades d'un objecte sf o sfc i estableix o substitueix el sistema de referència de coordenades recuperat de l'objecte.

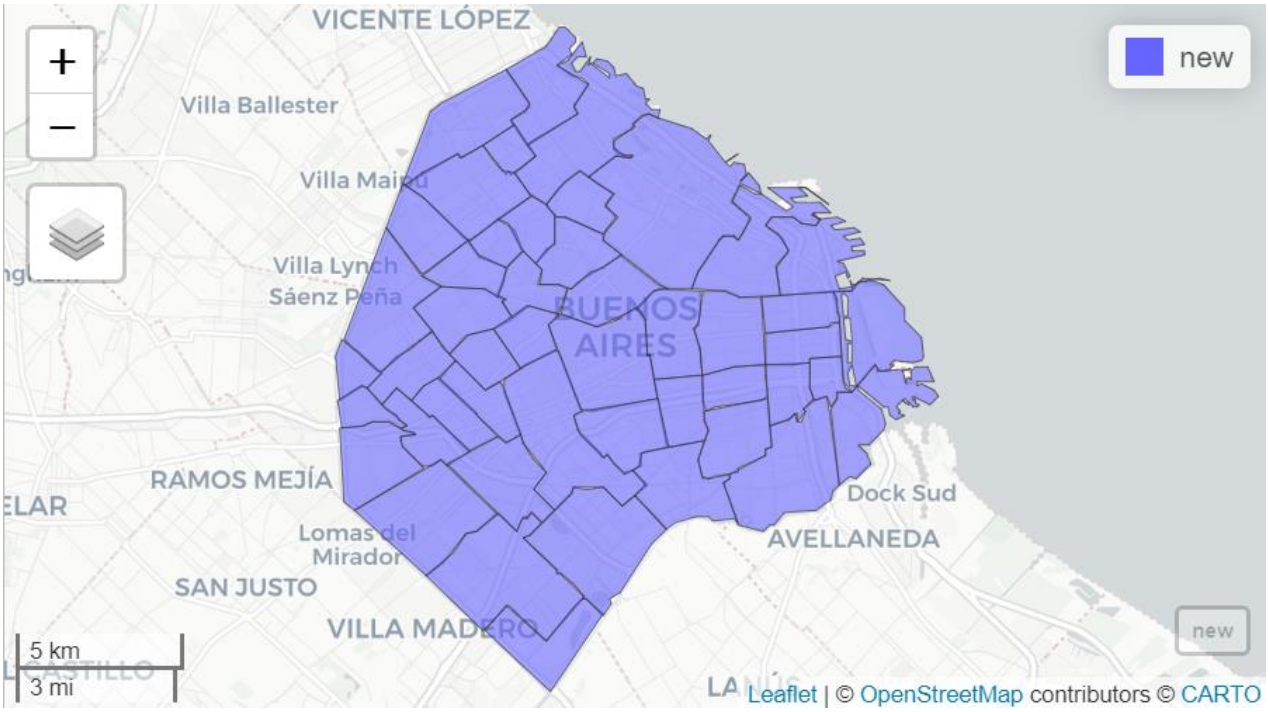
Instal·leu el paquet necessari i carregueu la llibreria

```
> install.packages("mapview")
```

```
>library(mapview)
```

Utilitzeu mapview amb les dades sf que heu creat en l'últim exercici.

```
mapview(new)
```



Podeu crear els vostres propis mapes interactius. Copieu per exemple:

```
d <- data.frame(
    place = c("London", "Paris", "Madrid", "Rome"),
```

```
    long = c(-0.118092, 2.349014, -3.703339, 12.496366),
    lat = c(51.509865, 48.864716, 40.416729, 41.902782),
    value = c(200, 300, 400, 600))
dsf <- st_as_sf(d, coords = c("long", "lat"))
st_crs(dsf) <- 4326
mapview(dsf)
```

Busca la longitud i latitud de Barcelona i afegeix les dades al mapa per value =500

```
d <- data.frame(
    place = c("London", "Paris", "Madrid", "Barcelona", "Rome"),
    long = c(-0.118092, 2.349014, -3.703339, 2.15899, 12.496366),
    lat = c(51.509865, 48.864716, 40.416729, 41.38879, 41.902782),
    value = c(200, 300, 400, 500, 600))
dsf <- st_as_sf(d, coords = c("long", "lat"))
st_crs(dsf) <- 4326
mapview(dsf)
```

Judit Chamorro Servent
Bellaterra, Abril 2025