

## MEMORIA DESCRIPTIVA – PRÁCTICA SPARK

Repositorio GitHub - <https://github.com/CarlosLopezPalau/Spark/>

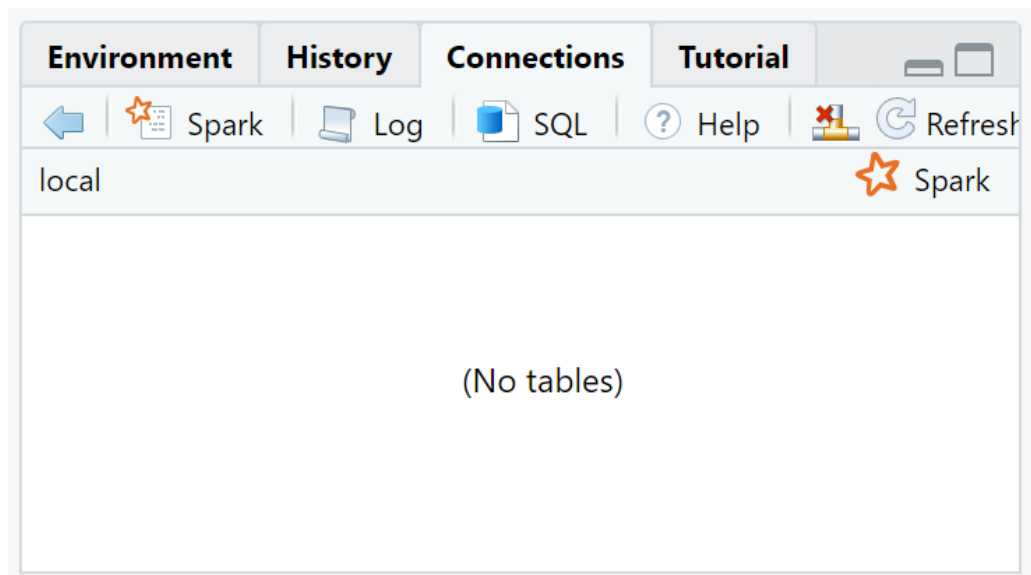
A lo largo de esta memoria voy a tratar de justificar mis respuestas de manera esquemática, incluyendo así las diferentes tablas y gráficos obtenidos.

*a) De fuentes veraces, lea los archivos que se indican en el anexo, como podrá apreciar, el/los archivos contienen miles de filas por decenas de columnas; solo es posible tratarlos utilizando Spark si queremos repuestas en tiempo real;*

```
1 #Práctica de Spark
2
3 #librerías
4 pacman::p_load(httr, tidyverse, leaflet, janitor, readr, sparklyr, XML, xlsx)
5
6 #spark
7 library(sparklyr)
8 library(dplyr)
9 sc <- spark_connect(master = "local")
10
11
```

En primer lugar, es fundamental instalar una serie de librerías, que usaremos posteriormente para realizar la práctica.

Como se ha comentado anteriormente, para tratar un dataset tan grande, necesitaremos emplear Spark, por lo que nos conectamos con el código anterior, a través de la librería sparklyr.



*i. Limpie el dataset (la información debe estar correctamente formateada, por ejemplo: lo que es de tipo texto no debe tener otro tipo que no sea texto, ponga el formato correcto en los números, etc.*

```

11
12 #código
13 url<- "https://sedeaplicaciones.minetur.gob.es/ServiciosRESTCarburantes/PreciosCarburantes/Estaci
14
15 htrr::GET(url)
16
17 ds <- jsonlite::fromJSON(url)
18 ds <- ds$ListaEESSPrecio
19 ds <- ds %>% as_tibble() %>% clean_names()
20
21 ds <- ds %>% type_convert(locale = locale(decimal_mark = ",")) %>% view() %>% clean_names()
22
23 view(ds)

```

Gracias a este código podemos obtener un archivo, a través de un link, que se va actualizando cada 'x' tiempo, por lo que es dinámico. Trataremos de limpiar las diferentes columnas y eliminar los duplicados. Además, gracias a 'clean\_names' también tendremos la posibilidad de establecer el formato correcto de cada variable.

Cabe destacar que también modificaremos el significado de la coma para el programa, de este modo seguirá el sistema 'común a nuestro país'.

*ii. genere un informe y explique si encuentra alguna anomalía, en el punto ii.*

Considero que la anomalía más destacable es la aparición de la coma (,) como indicador de número decimal en lugar del punto (.). Para solucionar este problema podríamos realizar un replace, no obstante, esto requiere de cierto gasto computacional innecesario. Por este motivo modificaremos la marca decimal del propio programa, como se ve en la imagen anterior.

*iii. Cree una nueva columna que deberá llamarse **low-cost**, y determine cual es el precio promedio de todos los combustibles a nivel comunidades autónomas, así como para las provincias, tanto para el territorio peninsular e insular, esta columna deberá clasificar las estaciones por lowcost y no lowcost.*

```

24
25 ds<- ds %>%mutate(lowcost=rotulo%in%c("REPSOL","CAMPASA","BP", "SHELL","GALP", "CEPSA")) %>% view()
26
27 #hacer mutate/replace para low_cost y no low cost renombrar
28 ds$lowcost[ds$lowcost == TRUE] <- 'No_Lowcost'
29 ds$lowcost[ds$lowcost == FALSE] <- 'Lowcost'
30

```

Gracias a la primera sentencia podemos crear una columna llena de TRUE en caso de incluir tales nombres o FALSE en caso contrario. Una vez tenemos esta columna simplemente sustituiremos 'TRUE' por 'No\_Lowcost' y 'FALSE' por 'Lowcost'.

```

32
33 media_total_comunidades <- ds %>% select(precio_bioetanol, precio_biodiesel, precio_gas_natural_compr
34 group_by(idccaa) %>%
35 summarise(media_precio_bioetanol=mean(precio_bioetanol, na.rm=TRUE), mean(precio_biodiesel, na.rm=
36 view()
37
38 media_total_provincias <- ds %>% select(precio_bioetanol, precio_biodiesel, precio_gas_natural_compr
39 group_by(provincia) %>%
40 summarise(media_precio_bioetanol=mean(precio_bioetanol, na.rm=TRUE), mean(precio_biodiesel, na.rm=
41 view()
42

```

En esta imagen no podemos apreciar el código completo, no obstante, seleccionaremos todos los precios de las diferentes gasolinas y posteriormente con el summarise realizaremos las diferentes medias. En el primer caso los agruparemos en base a la comunidad autónoma, mientras que en el segundo tendremos en cuenta la provincia.

	idccaa	media_precio_bioetanol	mean(precio biodiesel, na.rm = TRUE)	mean(precio gas natural comprimido, na.rm = TRUE)	mean(precio gas natural licuado, na.rm = TRUE)	mean(precio gas natural, na.rm = TRUE)
1	01	NaN	1.308444	1.440000	1.596000	
2	02	NaN	NaN	1.479750	1.449667	
3	03	1.729	NaN	1.466667	2.200000	
4	04	NaN	NaN	NaN	NaN	
5	05	NaN	NaN	0.920000	NaN	
6	06	NaN	1.301000	NaN	NaN	
7	07	NaN	NaN	1.327091	1.452667	
8	08	NaN	1.389000	1.314833	1.457000	
9	09	1.672	1.281308	1.573727	1.600615	
10	10	NaN	1.293500	1.625889	1.774100	
11	11	NaN	1.328000	1.513000	1.566333	

Por comunidad autónoma

	provincia	media_precio_bioetanol	mean(precio biodiesel, na.rm = TRUE)	mean(precio gas natural comprimido, na.rm = TRUE)	mean(precio gas natural licuado, na.rm = TRUE)	mean(precio gas natural, na.rm = TRUE)
1	ALBACETE	NaN	NaN	1.664500	1.815000	
2	ALICANTE	NaN	1.348000	1.472000	1.435333	
3	ALMERÍA	NaN	NaN	NaN	NaN	
4	ARABA/ÁLAVA	NaN	1.417667	0.948000	0.993000	
5	ASTURIAS	1.729	NaN	1.466667	2.200000	
6	ÁVILA	NaN	1.399000	NaN	NaN	
7	BADAJOS	NaN	NaN	1.670000	1.655000	
8	BALEARS (ILLES)	NaN	NaN	NaN	NaN	
9	BARCELONA	1.672	1.275250	1.634667	1.616833	
10	BIZKAIA	NaN	NaN	1.710000	2.200000	
11	BURGOS	NaN	NaN	1.123333	1.102500	

Por provincia

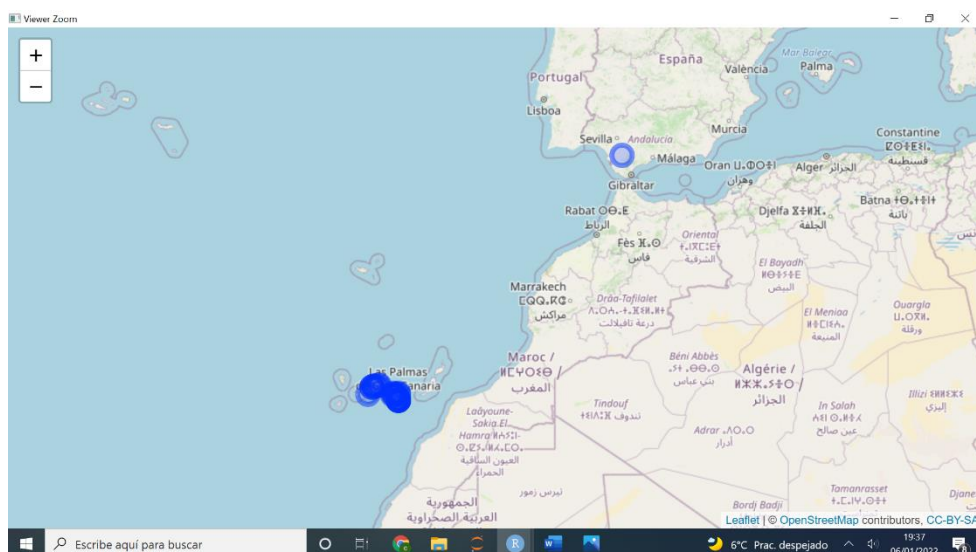
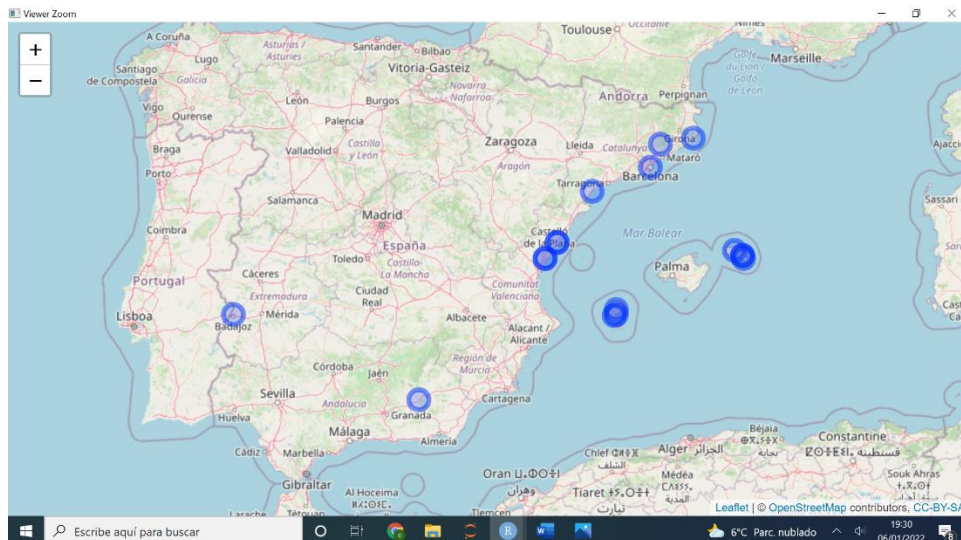
iv. Imprima en un mapa interactivo, la localización del top 10 mas caras y otro mapa interactivo del top 20 mas baratas, estos 2 archivos deben guardarse en formato HTML y un pdf para su posterior entrega al inversor, nombre de los archivos: top\_10.html, top\_10.pdf y top\_20.html, top\_20.pdf.

```

42
43 #iv. Imprima en un mapa interactivo, la localización del top 10 mas caras y otro mapa interactivo
44 ds %>% select(rotulo, latitud, longitud_wgs84, precio_gasolina_95_e5, localidad, direccion) %>%
45   top_n(10, precio_gasolina_95_e5) %>%
46   leaflet() %>% addTiles() %>%
47   addCircleMarkers(lng = ~longitud_wgs84, lat = ~latitud, popup = ~rotulo, label = ~precio_gasolina_95_e5)
48
49 ds %>% select(rotulo, latitud, longitud_wgs84, precio_gasolina_95_e5, localidad, direccion) %>%
50   top_n(-20, precio_gasolina_95_e5) %>% leaflet() %>% addTiles() %>% addCircleMarkers(lng = ~longitud_wgs84, lat = ~latitud, popup = ~rotulo, label = ~precio_gasolina_95_e5)
51 lowcost_22176323 <- ds
52 View(lowcost_22176323)
53

```

Combinando tanto la función top\_n, que nos indica los precios más elevados, como la leaflet que nos permite realizar ‘gráficos en un mapa’, obtendremos los siguientes resultados: (cabe destacar que para obtener el top20 más baratas simplemente indicamos el valor de manera negativa)



v. conseguidos los objetivos anteriores, debe guardar este “archivo” en una nueva tabla llamada `low-cost_num_expediente` y deberá estar disponible también en su repositorio de Github con el mismo nombre y formato csv.

```
51
52 lowcost_22176323 <- ds
53 View(lowcost_22176323)
54 write.csv(lowcost_22176323,"C:/Users/ho1a/Desktop/Spark/lowcost_22176323.csv", row.names = FALSE)
55
56 #R.Este empresario tiene sus residencias habituales en Madrid y Barcelona . por lo que, en prin
```

En este caso guardaremos las modificaciones realizadas llamándolas ‘low-cost\_num\_22176323’. Posteriormente lo ‘exportaremos’ a nuestro pc en csv, para posteriormente subirlo al repositorio de Github.

b) Este empresario tiene sus residencias habituales en Madrid y Barcelona, por lo que, en principio le gustaría implantarse en cualquiera de las dos antes citadas, y para ello quiere saber:

i. Cuántas gasolineras tiene la comunidad de Madrid y la comunidad de Cataluña, cuantas son low-cost, cuantas no lo son.

```
#i. cuántas gasolineras tiene la comunidad de Madrid y en
MyB <- ds %>% select(idcaa, lowcost, provincia) %>%
  filter(idcaa=='13'|idcaa=='09') %>%
  group_by(idcaa) %>% count(lowcost) %>%
  View()
```

Filtramos teniendo en cuenta que el `idcaa=09` pertenece a Cataluña y la `idcaa=13` pertenece a la Comunidad de Madrid. Agrupamos en base a la comunidad y realizamos un recuento.

El resultado es el siguiente:

	idcaa	lowcost	n
1	09	Lowcost	826
2	09	No_Lowcost	653
3	13	Lowcost	316
4	13	No_Lowcost	474

ii. Además, necesita saber cuál es el precio promedio, el precio más bajo y el más caro de los siguientes carburantes: gasóleo A, y gasolina 95 e Premium.

```
67 |
68 | #ii. además, necesita saber cuál es el precio promedio, el precio más bajo y el más caro de l
69 | informe_MAD_BCN_22176323 <- ds %>% select(idccaa, lowcost, provincia, precio_gasoleo_a, precio_gasolina_95_
70 | filter(idccaa=="09" | idccaa=="13") %>%
71 | drop_na() %>%
72 | group_by(idccaa, lowcost) %>%
73 | summarise(max(precio_gasoleo_a), min(precio_gasoleo_a), mean(precio_gasoleo_a), max(precio_gasolina_95_e5
74 | view(informe_MAD_BCN_22176323)
75 |
```

Teniendo en cuenta los argumentos fijados anteriormente, seleccionaremos los diferentes precios de los carburantes que queremos analizar. Posteriormente los agruparemos según comunidad y si es lowcost o no lo es. La parte ‘principal’ del código se trata del summarise donde incluiremos la función max, mean y min.

Obtendremos una tabla como la siguiente (recortada):

	idccaa	lowcost	max(precio_gasoleo_a)	min(precio_gasoleo_a)	mean(precio_gasoleo_a)	max(precio_gasolina_95_e5_premium)	min(precio_gasolina_95_e5_premium)
1	09	Lowcost	1.449	1.189	1.317850	1.659	
2	09	No_Lowcost	1.469	1.319	1.411891	1.705	
3	13	Lowcost	1.449	1.389	1.422333	1.659	
4	13	No_Lowcost	1.459	1.339	1.425873	1.689	

iii. Conseguido el objetivo, deberá guardar este “archivo” en una nueva tabla llamada informe\_MAD\_BCN\_expediente y deberá estar disponible también en su repositorio con el mismo nombre en formato csv.

```
77 |
78 | #iii. Conseguido el objetivo, deberá guardar este "archivo" en una nueva tabla llamada informe_MAD_BCN_exp
79 | write.csv(informe_MAD_BCN_22176323, "C:/Users/hola/Desktop/spark/informe_MAD_BCN_22176323.csv", row.names =
80 |
81 | #c. Por si las comunidades de Madrid y Cataluña no se adapta a sus requerimientos, el empresari
82 |
83 | #i. conocer a nivel municipios cuántas gasolineras son low cost, cuántas no lo son, cuál es el precio
```

Convertimos esta tabla a csv y la almacenamos en nuestro pc, mediante la función ‘write.csv’. Será necesario seleccionar la ruta de almacenamiento.

c) Por si las comunidades de Madrid y Cataluña no se adaptan a sus requerimientos, el empresario también quiere:

i. Conocer a nivel municipios, cuántas gasolineras son low-cost, cuantas no lo son, cual es el precio promedio, el precio más bajo y el más caro de los siguientes carburantes: gasóleo A, y gasolina 95 e5 Premium, en todo el TERRITORIO NACIONAL, exceptuando las grandes CIUDADES ESPAÑOLAS (“MADRID”, “BARCELONA”, “SEVILLA” y “VALENCIA”)



```

83 #i. conocer a nivel municipios, cuántas gasolineras son low-cost, cuántas no lo son, cuál es el precio
84 informe_no_grandes_ciudades_22176323 <- ds %>% select(idccaa, id_municipio, municipio, lowcost, precio_gasoleo_a)
85 group_by(municipio, lowcost) %>%
86 filter(!municipio %in% c("Madrid", "Barcelona", "Sevilla", "Valencia")) %>%
87 summarise(max(precio_gasoleo_a), min(precio_gasoleo_a), mean(precio_gasoleo_a), max(precio_gasolina_95_e5))
88 view(informe_no_grandes_ciudades_22176323)
89
90 Cantidad_gasolineras <- ds %>% select(id_municipio, municipio, lowcost) %>%
91 filter(!municipio %in% c("Madrid", "Barcelona", "Sevilla", "Valencia")) %>%
92 group_by(municipio) %>%
93 count(lowcost)
94 view(Cantidad_gasolineras)
95

```

En un principio traté de realizar este apartado incluyendo el count en la sentencia ‘global’, no obstante, me aparecían diversos errores y decidí realizarlo por separado.

La parte diferenciadora de este ejercicio es la necesidad de emplear un filter junto al signo de exclamación ‘!’, que implica filtrar incluyendo todos los municipios que no cumplan esos parámetros. Además, como queremos realizar un estudio a nivel municipal será fundamental realizar un groupby en base a la variable municipio.

De este modo obtenemos las siguientes tablas:

	municipio	lowcost	max(precio_gasoleo_a)	min(precio_gasoleo_a)	mean(precio_gasoleo_a)	max(precio_gasolina_95_e5_premium)
1	Abadín	Lowcost	1.429	1.429	1.429000	
2	Abadín	No_Lowcost	1.409	1.409	1.409000	
3	Abadiño	Lowcost	1.359	1.315	1.337000	
4	Abanilla	Lowcost	1.319	1.299	1.309000	
5	Abanilla	No_Lowcost	1.399	1.399	1.399000	
6	Abanto y Ciérvana-Abanto Zierbena	Lowcost	1.459	1.449	1.455667	
7	Abanto y Ciérvana-Abanto Zierbena	No_Lowcost	1.449	1.449	1.449000	
8	Abarán	Lowcost	1.379	1.359	1.369000	
9	Abarán	No_Lowcost	1.359	1.359	1.359000	
10	Abegondo	Lowcost	1.278	1.278	1.278000	
11	Abegondo	No_Lowcost	1.409	1.409	1.409000	
12	Abejar	No_Lowcost	1.379	1.379	1.379000	

Showing 1 to 12 of 4,629 entries, 8 total columns

	municipio	lowcost	n
1	Abadín	Lowcost	1
2	Abadín	No_Lowcost	1
3	Abadiño	Lowcost	2
4	Abanilla	Lowcost	2
5	Abanilla	No_Lowcost	1
6	Abanto y Ciérvana-Abanto Zierbena	Lowcost	3
7	Abanto y Ciérvana-Abanto Zierbena	No_Lowcost	1
8	Abarán	Lowcost	2
9	Abarán	No_Lowcost	1
10	Abegondo	Lowcost	1
11	Abegondo	No_Lowcost	1
12	Abejar	No_Lowcost	1
13	Abengibre	Lowcost	1

Showing 1 to 13 of 4,629 entries, 3 total columns

ii. Conseguido el objetivo, deberá guardar este “archivo” en una nueva tabla llamada `informe_no_grandes_ciudades_expediente` y deberá estar disponible también en su repositorio con el mismo nombre en formato Excel.

```

95
96 #ii. Conseguido el objetivo, deberá guardar este "archivo" en una nueva tabla llamada informe_no_grandes_c
97 install.packages("writexl")
98 library("writexl")
99
100 writexl(informe_no_grandes_ciudades_22176323,"C:/Users/hola/Desktop/Spark/informe_no_grandes_ciudades_22
101

```

En este caso, necesitaremos instalar ‘writexl’ para poder guardar la tabla en formato Excel.

d. Determine:

i. Qué gasolineras se encuentran abiertas las 24 horas exclusivamente, genere una tabla llamada `no_24_horas` sin la variable `horario` (es decir no debe aparecer esta columna).

```

101
102 #D. Determine :
103 #i. que gasolineras se encuentran abiertas las 24 h
104 no_24_horas <- ds[(ds$horario == 'L-D: 24H'),]
105 no_24_horas <- no_24_horas[,-3]
106 View(no_24_horas)
107

```

Se trata de un apartado bastante sencillo, simplemente debemos quedarnos con aquellas gasolineras que abran de L-D las 24 horas del día. Una vez establecida esta condición, simplemente eliminaremos mediante su índice la columna ‘horario’.

Este apartado puede resolverse también a través de un filter. La tabla obtenida es la siguiente:

c.p	direccion	latitud	localidad	longitud wgs84	margen	municipio	pre
1 02001	CALLE FEDERICO GARCIA LORCA, 1	39.00086	ALBACETE	-1.849833	D	Albacete	
2 02005	AVENIDA MENÉNDEZ PIDAL, 58	39.00333	ALBACETE	-1.864917	N	Albacete	
3 02006	CARRETERA JAEN KM. 2	38.99467	ALBACETE	-1.871722	D	Albacete	
4 02006	CR N- 301, 244	39.02778	ALBACETE	-1.892972	I	Albacete	
5 02004	CARRETERA N-322 KM. 349	38.97017	ALBACETE	-1.919778	D	Albacete	
6 02099	CTRA. AGUAS NUEVAS KM 0,25	38.87028	SALOBRRAL (EL)	-1.922722	D	Albacete	
7 02007	AVENIDA PRIMERA, S/N	39.02214	ALBACETE	-1.882056	D	Albacete	
8 02007	CALLE C/ AUTOVIA C/V AVO. SEXTA, S/N	39.02958	ALBACETE	-1.888083	D	Albacete	
9 02007	AVENIDA POLIGONO CAMPOLLANO AVENIDA 0, 67	39.02158	ALBACETE	-1.886556	N	Albacete	
10 02007	POLIGONO AVDA. 3ª, ESQUINA C/AUTOVIA, 69	39.01800	ALBACETE	-1.874333	D	Albacete	
11 02007	POLIGONO CAMPOLLANO C/ F, 13	39.01247	ALBACETE	-1.889417	D	Albacete	
12 02007	AVENIDA 3ª ESQUINA C/ F Nº3, 3	39.00942	ALBACETE	-1.885694	N	Albacete	

Showing 1 to 12 of 4,756 entries, 32 total columns

Se puede ver una menor cantidad de filas, ya que nos hemos desecho de gran cantidad de gasolineras.



*ii. Conseguido el objetivo, deberá guardar este “archivo” en una nueva tabla llamada no\_24\_horas y deberá estar disponible también en su repositorio con el mismo nombre en formato Excel.*

```
106 View(no_24_horas)
107
108 #ii. Conseguido el objetivo, deberá guardar este "archivo" en una nueva tabla llamada no_24_horas y deberá
109 write_xlsx(no_24_horas, "C:/Users/hola/Desktop/Spark/no_24_horas_22176323.xlsx")
110
111 #E. Uno de los factores más importantes para que el empresario se decante a instalar nuevas gasolineras es
```

Tal y como hemos hecho en apartados anteriores, simplemente descargaremos la tabla en formato Excel.

*e) Uno de los factores más importantes para que el empresario se decante a instalar nuevas gasolineras es la demanda que viene dada por la población y la competencia existente en un municipio donde se pretenda implantar las gasolineras, para responder a esta pregunta de negocio,*

*i. deberá añadir la población al dataset original creando una nueva columna denominada población, esta información debe ser veraz y la más actualizada, la población debe estar a nivel municipal (todo el territorio nacional)*

```
113 #i. deberá añadir la población al dataset original creando una nueva columna denominada población,
114 library(readxl)
115 pobmun21 <- read_excel("pobmun21.xlsx", skip = 2)%>% drop_na() %>% clean_names()
116
117 poblacion <- rename(pobmun21, id_provincia = 'cpro', id_municipio='cmun', municipio='nombre')
118
119 union <- left_join(ds, poblacion, 'municipio')
120 View(union)
```

En primer lugar, debemos leer el Excel descargado del INE, donde encontramos el registro municipal de toda España. Mediante `clean_names`, limpiaremos el dataset. Antes de realizar el join, es fundamental que la variable ‘key’ coincida en cuanto a nombre. En este caso modificaremos ‘nombre’ por ‘municipio’ para que ambos datasets coincidan.

Una vez tenemos todo preparado realizaremos un left join entre el dataset original y la población, para de este modo obtener una columna que indique el censo de cada municipio.

La forma óptima de resolver este apartado era unir ambos datasets a través del `id_municipio`, ya que nos evitaríamos problemas con los idiomas o signos en los nombres de los municipios. No obstante, traté de realizarlo de tal modo y R mostraba diversos fallos que no me permitían continuar.

El resultado final es esta tabla, que incluye la población

percent_ester_metilico	ideess	id_municipio.x	id_provincia.x	idcaa	lowcost	id_provincia.y	provincia.y	id_municipio.y	pob21
0	4375	52	02	07	Lowcost	02	Albacete	001	748
0	5122	53	02	07	No_Lowcost	02	Albacete	002	496
0	10765	54	02	07	Lowcost	02	Albacete	003	172722
0	4438	54	02	07	No_Lowcost	02	Albacete	003	172722
0	13933	54	02	07	Lowcost	02	Albacete	003	172722
0	12054	54	02	07	Lowcost	02	Albacete	003	172722
0	5195	54	02	07	No_Lowcost	02	Albacete	003	172722
0	4369	54	02	07	Lowcost	02	Albacete	003	172722
0	14123	54	02	07	No_Lowcost	02	Albacete	003	172722
0	15000	54	02	07	Lowcost	02	Albacete	003	172722
0	5313	54	02	07	No_Lowcost	02	Albacete	003	172722
0	4415	54	02	07	Lowcost	02	Albacete	003	172722

Showing 1 to 12 of 11,432 entries, 37 total columns

ii. este empresario ha visto varios sitios donde potencialmente le gustaría instalar su gasolinera, esos sitios están representados por la dirección, desde esta última calcule cuanta competencia (nombre de la gasolinera y dirección) tiene en:

1. En un radio de 1km (genere mapa\_competencia1.html)
2. En un radio de 2km (genere mapa\_competencia2.html)
3. En un radio de 4km (genere mapa\_competencia3.html)

No comprendo el enunciado, no se establece una dirección inicial a partir de la cual establecer el radio.

iii. genere el TopTen de municipios entre todo el territorio nacional excepto el territorio insular, donde no existan gasolineras 24horas, agrupadas entre low-cost y no low-cost, deberá guardar este “archivo” en una nueva tabla llamada informe\_top\_ten\_expediente y deberá estar disponible también en su repositorio con el mismo nombre en formato csv.

```

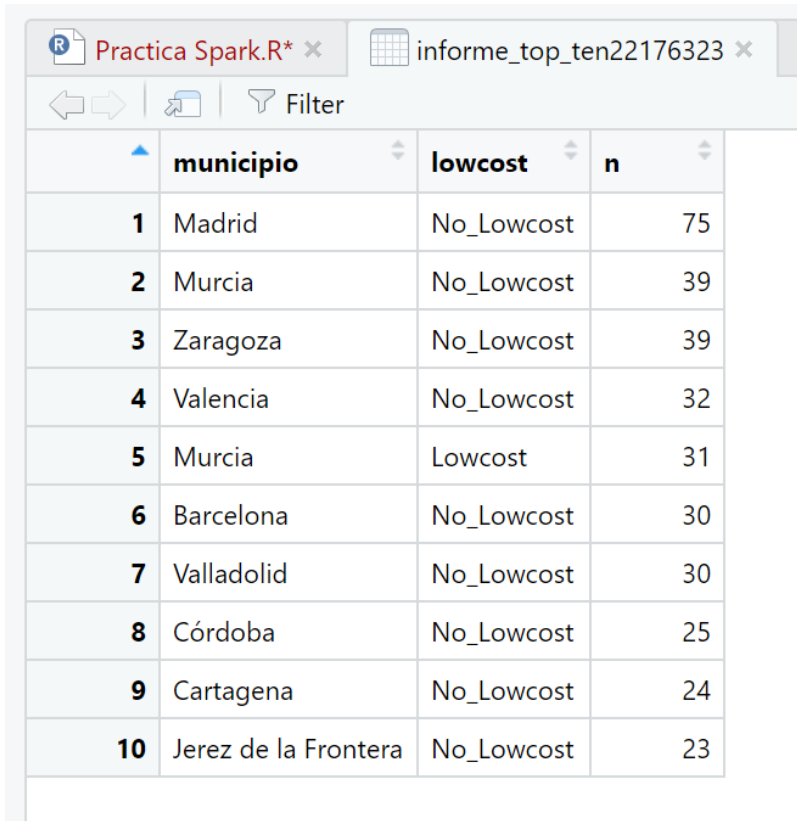
133 #iii. genere el TopTen de municipios entre todo el territorio nacional excepto el territorio insul
134 informe_top_ten22176323 <- union %>%
135   filter(!provincia.x%in%c('BALEARS (ILLES)', 'PALMAS (LAS)')) %>%
136   filter(!horario == 'L-D: 24H') %>%
137   group_by(municipio, lowcost) %>%
138   count()
139 informe_top_ten22176323 <- informe_top_ten22176323[order(informe_top_ten22176323$n, decreasing = TRUE),]
140 informe_top_ten22176323 <- head(informe_top_ten22176323, 10)
141 View(informe_top_ten22176323)

```

En el filtro eliminaremos el territorio insular y las gasolineras 24 horas. Además, realizaremos un groupby según la variable municipio y lowcost, ya que se trata de un top municipal. También realizaremos un cont.

Por último, lo ordenaremos de manera descendente y seleccionaremos únicamente los diez primeros valores.

El resultado es el siguiente:



	municipio	lowcost	n
1	Madrid	No_Lowcost	75
2	Murcia	No_Lowcost	39
3	Zaragoza	No_Lowcost	39
4	Valencia	No_Lowcost	32
5	Murcia	Lowcost	31
6	Barcelona	No_Lowcost	30
7	Valladolid	No_Lowcost	30
8	Córdoba	No_Lowcost	25
9	Cartagena	No_Lowcost	24
10	Jerez de la Frontera	No_Lowcost	23

Guardamos la tabla en formato csv:

```
141 view(informe_top_ten22176323)
142 write.csv(informe_top_ten22176323,"C:/Users/hola/Desktop/Spark/informe_top_ten22176323.csv", row.names = FALSE)
143
```