2. Preparación para modelización

Taller: calidad del aire

Minería de datos II

Curso 2019/2020

Contents

1 Entorno

Crea un **script R** en la carpeta `taller_calidad_aire/src` que se llame `02_preparacion_modelizacion.R`; será el script que desarrollaremos en este documento.

Al principio del script, carga el paquete `tidyverse`:

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts ------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

2 Importación de datos

Para comenzar a trabajar, necesitamos importar los datos que obtuvimos de la fase anterior. Como lo que guardamos fue un archivo `.RDS`, los datos ya serán un `data.frame` de R y, por tanto, no tenemos que preocuparnos del formato de los datos como teníamos que hacerlo con un `.csv`.

```
calidad_aire <- readRDS("taller_calidad_aire/data/01_transformacion.RDS")
```

3 Análisis exploratorio de datos

Antes de realizar cualquier transformación sobre los datos, necesitamos conocerlos en profundidad. En cualquier proyecto de minería de datos, una de las primeras fases una vez que los datos se han importado correctamente es hacer un **análisis exploratorio de datos** (*EDA*).

Un paquete que nos puede ayudar a esto es el paquete `skimr`. Este paquete nos da un resumen de los datos.

```
skimr::skim(calidad_aire)
```

Table 1: Data summary

Name	calidad_aire
Number of rows	1064
Number of columns	18
Column type frequency:	
Date	1
numeric	17
Group variables	None

Variable type: Date

skim_variable	n_missing	complete_rate	min	max	median	n_unique
fecha	0	1	2017-01-01	2019-11-30	2018-06-16	1064

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
ano	0	1	2017.97	0.81	2017.00	2017.00	2018.00	2019.00	2019.00
mes	0	1	6.37	3.37	1.00	3.00	6.00	9.00	12.00
dia	0	1	15.71	8.80	1.00	8.00	16.00	23.00	31.00
so2	0	1	7.28	2.49	2.50	5.30	7.00	8.57	18.20
co	0	1	0.35	0.13	0.14	0.26	0.31	0.40	1.00
no	0	1	18.70	23.97	1.58	5.12	8.69	21.01	171.75
no2	0	1	37.44	17.34	8.83	24.54	33.56	47.26	104.33
pm25	0	1	9.93	5.14	2.40	6.00	9.20	12.68	40.33
pm10	0	1	18.58	10.37	2.58	10.98	17.00	23.55	144.17
nox	0	1	66.11	52.38	12.08	32.42	47.34	79.29	361.83

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
o3	0	1	51.24	22.99	2.57	33.68	54.82	70.11	105.36
tol	0	1	2.11	1.72	0.20	1.01	1.60	2.55	13.52
ben	0	1	0.49	0.33	0.12	0.27	0.38	0.60	2.20
ebe	0	1	0.46	0.51	0.10	0.18	0.28	0.50	3.45
tch	0	1	1.43	0.12	1.11	1.35	1.41	1.48	2.05
ch4	0	1	1.33	0.10	0.98	1.27	1.32	1.37	1.91
nmhc	0	1	0.10	0.04	0.01	0.07	0.10	0.13	0.31

> **Nota:** recuerda que al escribir `skimr::skim` le estamos diciendo a R que utilice la función `skim` que pertence al paquete `skimr`. Esto es útil cuando queremos utilizar la función `skim` una única vez y, por lo tanto, no hace falta hacer `library(skimr)`.

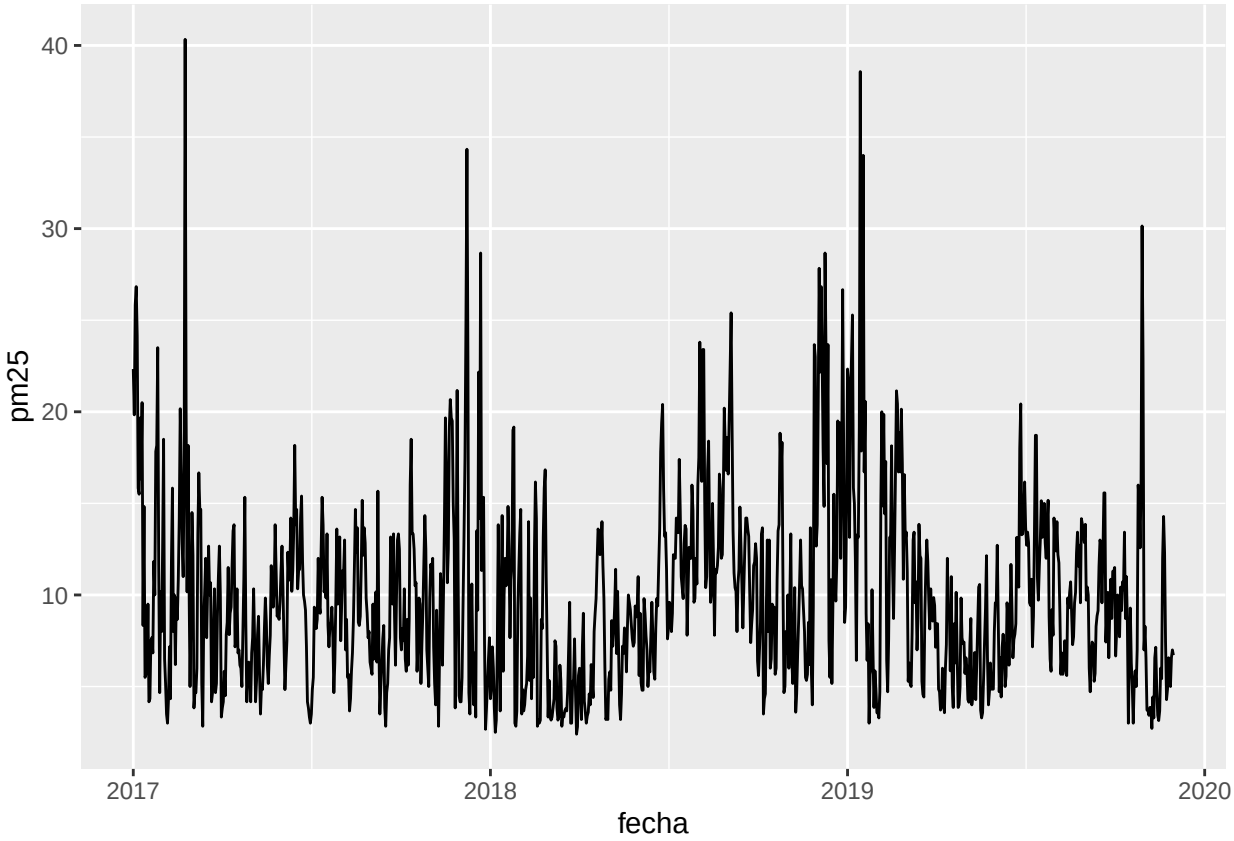
Si te fijas en la columan `m_missing` del resultado anterior, sabemos que no tenemos ningún dato ausente en ninguna variable. Por lo tanto, no hará falta recurrir a la imputación de valores ausentes.

En el apartado de la variable fecha de la salida anterior podemos ver que los datos van desde el `2017-01-01` hasta `2019-11-30`.

La variable que queremos predecir es `pm25`, así que vamos a estudiarla en más detalle.

3.1 Variable `pm25`

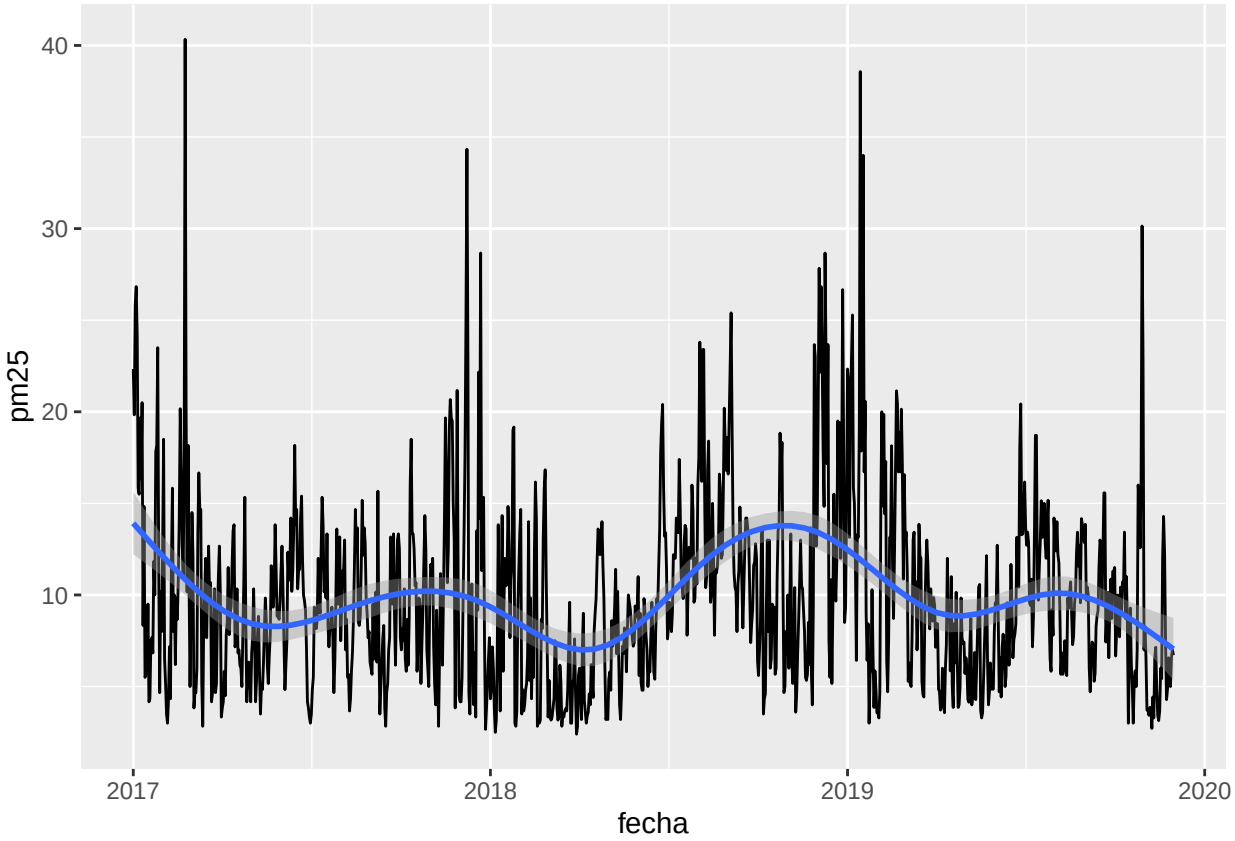
```
ggplot(data = calidad_aire,
       aes(x = fecha, y = pm25)) +
  geom_line()
```



En el gráfico anterior de puede apreciar un patrón *ondulante* que parece repetirse anualmente. Podemos verlo de forma más clara si añadimos la capa `geom_smooth()`

```
ggplot(data = calidad_aire,
       aes(x = fecha, y = pm25)) +
  geom_line() +
  geom_smooth()
```

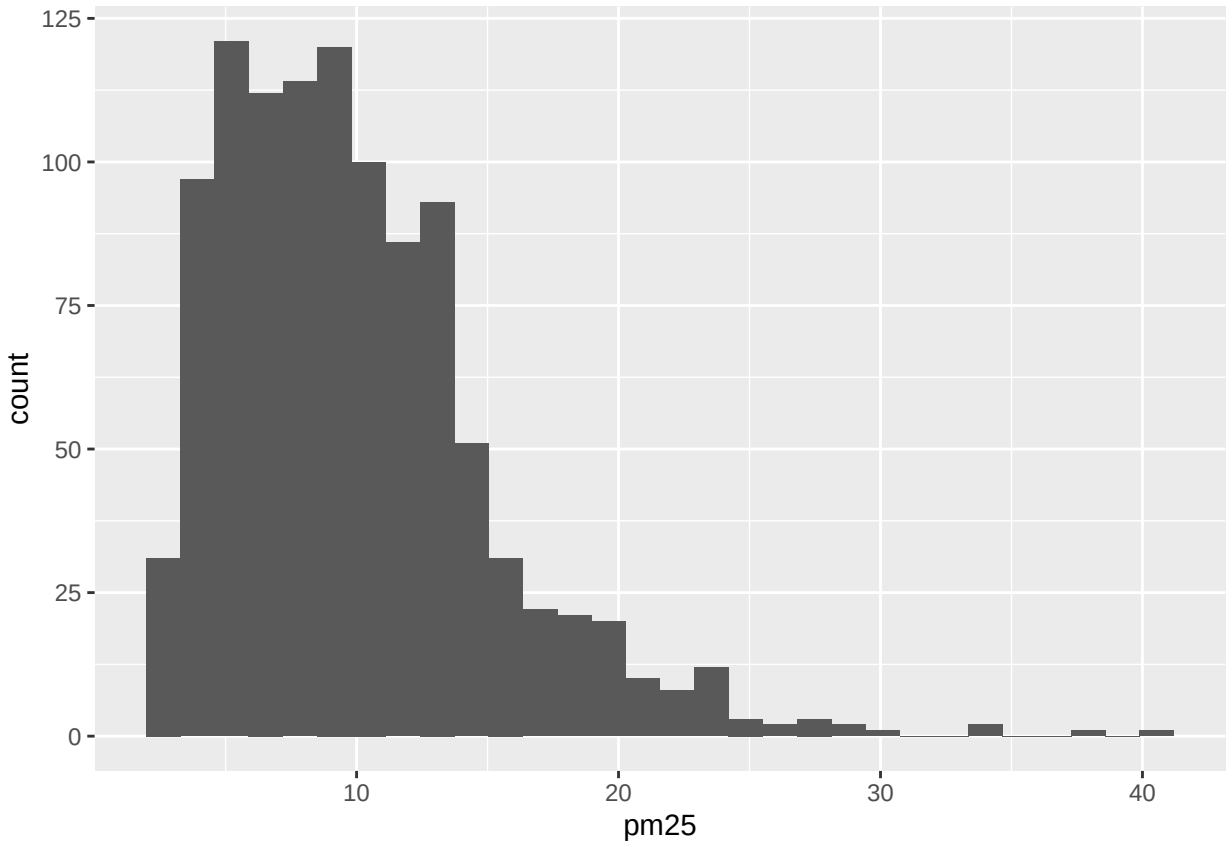
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



La distribución de la variable podemos verla mediante un histograma:

```
ggplot(data = calidad_aire,
       aes(x = pm25)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



4 Construcción de variables

Recordemos que el objetivo es **predecir el valor de `pm25` en el día posterior**. Eso significa que, si queremos predecir el valor para el 9 de febrero, tenemos que suponer que **solamente conocemos la información hasta el día 8 de febrero**. Por lo tanto, el valor que podremos utilizar de las variables para predecir un día, debe ser la información disponible hasta el día anterior. Por ejemplo, no podemos utilizar el valor de `so2` en el mismo día que el valor de `pm25` que queremos predecir. Necesitamos que el valor de cada variable esté *retrasado* en un día. Esto lo podemos hacer mediante la función `lag`. Para entenderlo, vamos hacer primero un ejemplo. Si tuviésemos el vector `c(1,2,3,4)`, si aplicamos la función `lag` obtendríamos

```
lag(c(1,2,3,4))
```

```
## [1] NA  1  2  3
```

Obviamente, el valor anterior del primer elemento del vector es desconocido y por eso aparece como `NA`.

Antes de utilizar la función `lag` debemos asegurarnos de que los datos estén ordenados de menor a mayor por la variable fecha, porque en caso contrario no tendría sentido lo que estaríamos haciendo:

```
calidad_aire <- arrange(calidad_aire, fecha)
```

Y ahora generamos una nueva variable `_lag` por cada variable que tengamos que retrasar. Por ejemplo

```
calidad_aire$so2_lag <- lag(calidad_aire$so2)
```

Este procedimiento habría que repetirlo demasiadas veces y sería demasiado pesado para hacerlo de forma manual. Para *automatizarlo* podemos utilizar la función `mutate_at`:

```
calidad_aire <- calidad_aire %>%
  mutate_at(vars(so2:nmhc), list(lag = lag))
```

Nota: lo que acabamos de hacer se puede traducir como: aplica la función `lag` a aquellas variables que están entre `so2` y `nmhc`. Al utilizar `list(lag = lag)`, cada variable que se crea termina en `_lag`.

Para finalizar la creación de estas variables, debemos eliminar todas las variables que no son lag y quedarnos solamente con `pm25` que es la variable que queremos predecir.

```
calidad_aire <- calidad_aire %>%
  select(fecha, ano:dia, so2_lag:nmhc_lag, pm25)
```

Es interesante conocer la correlación de la variable objetivo con respecto a las predictoras:

```
cor(calidad_aire$pm25,
    select(calidad_aire, ends_with("lag")),
    use = "complete.obs"
    )
```

```
##        so2_lag    co_lag    no_lag   no2_lag  pm25_lag  pm10_lag   nox_lag
## [1,] 0.4284642 0.5468669 0.4825432 0.5294305 0.6936286 0.5976506 0.5138496
##          o3_lag   tol_lag   ben_lag ebe_lag   tch_lag   ch4_lag  nmhc_lag
## [1,] -0.2627425 0.4335614 0.4198621  0.1981 0.4062614 0.3397541 0.3181111
```

> **Nota**: en la correlación utilizamos `use = "complete.obs"` para que no tenga en cuenta los `NA` en el cálculo.

Puedes ver que la mayor correlación de `pm25` se da con `pm25_lag`.

5 Train y test

Por último, como hacemos habitualmente, vamos a dividir el conjunto de datos en `train` y `test`. Entrenaremos con datos hasta `2019-09-01` y los restantes para `test`:

```
train <- calidad_aire[calidad_aire$fecha < as.Date("2019-09-01"),]
test <- calidad_aire[calidad_aire$fecha >= as.Date("2019-09-01"),]
```

6 Exportación de la información

Igual que hicimos en la fase anterior, alamacenamos estos datos en el disco duro.

```r
saveRDS(train, file = "taller_calidad_aire/data/train.RDS")
saveRDS(test, file = "taller_calidad_aire/data/test.RDS")
```