

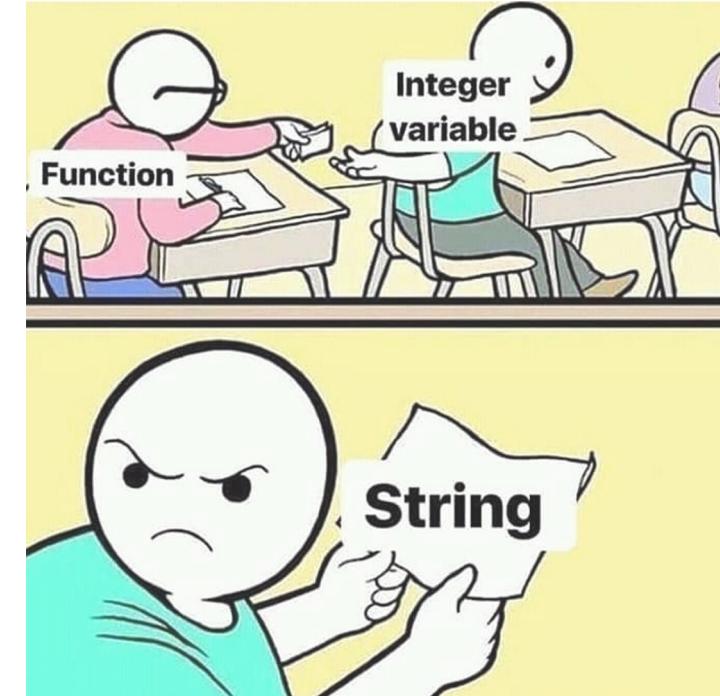


Factores, texto y fechas

{forcats, stringr, lubridate}

José Luis Texcalac Sangrador

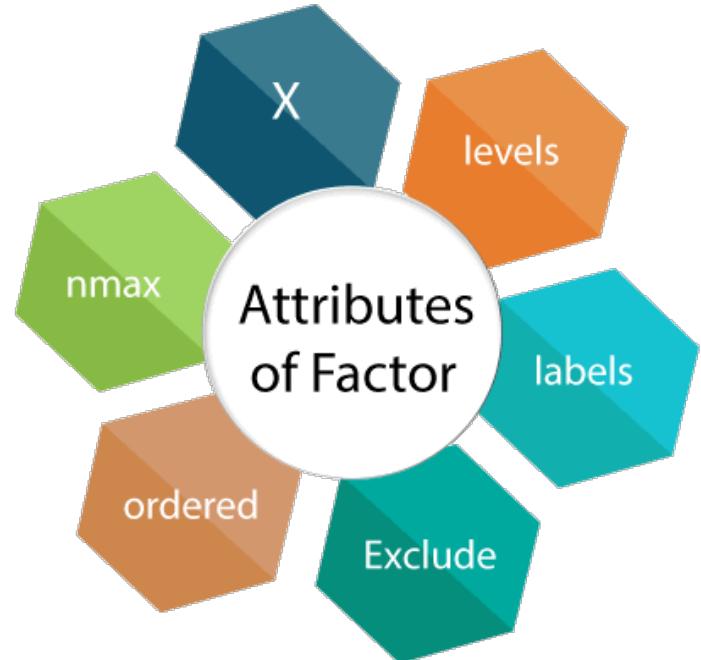
Procesamiento y visualización de datos espaciales en R





Factores en R

- Se usan para trabajar con variables categóricas, es decir, variables que tienen un conjunto fijo y conocido de valores posibles.
- Son útiles cuando quieras mostrar vectores de caracteres en un orden no alfabético.
- Los valores que un factor puede contener están delimitados por los niveles, es decir, las diferentes categorías presentes en una variable.
- Los niveles pueden estar codificados tanto en valores numéricos como caracteres.





Factores

malla

nombre	ojos	pelo
Luisa	azules	1
Juana	negros	2
Petra	azules	2
Maria	verdes	NA
Andrea	negros	1

summary(malla)

```
nombre          ojos          pelo
Length:5       Length:5       Min.  :1.0
Class :character Class :character 1st Qu.:1.0
Mode  :character Mode  :character Median :1.5
                           Mean   :1.5
                           3rd Qu.:2.0
                           Max.   :2.0
                           NA's    :1
```

Vector o variable a categorizar

Etiquetas que toma la variable

factor(variable, levels, labels)

Niveles que toma la variable



Valores distintos en un vector o variable

```
unique(malla$ojos)
```

```
[1] "azules" "negros" "verdes"
```

```
unique(malla$pelo)
```

```
[1] 1 2 NA
```



Factores

Vector o variable
a categorizar

Niveles que
toma la variable

Etiquetas que
toma la variable

```
factor(variable, levels, labels)
```

malla

nombre	ojos	pelo	ojos_f	pelo_f
Luisa	azules	1	azules	lacio
Juana	negros	2	negros	rizado
Petra	azules	2	azules	rizado
Maria	verdes	NA	verdes	NA
Andrea	negros	1	negros	lacio

```
malla <-  
  malla %>%  
  mutate(ojos_f = factor(ojos,  
                        levels = c("azules", "negros",  
                                  "verdes", "grises")),  
        pelo_f = factor(pelo,  
                        levels = c(1, 2),  
                        labels = c("lacio", "rizado")))) %>%  
  print()
```

summary(malla)

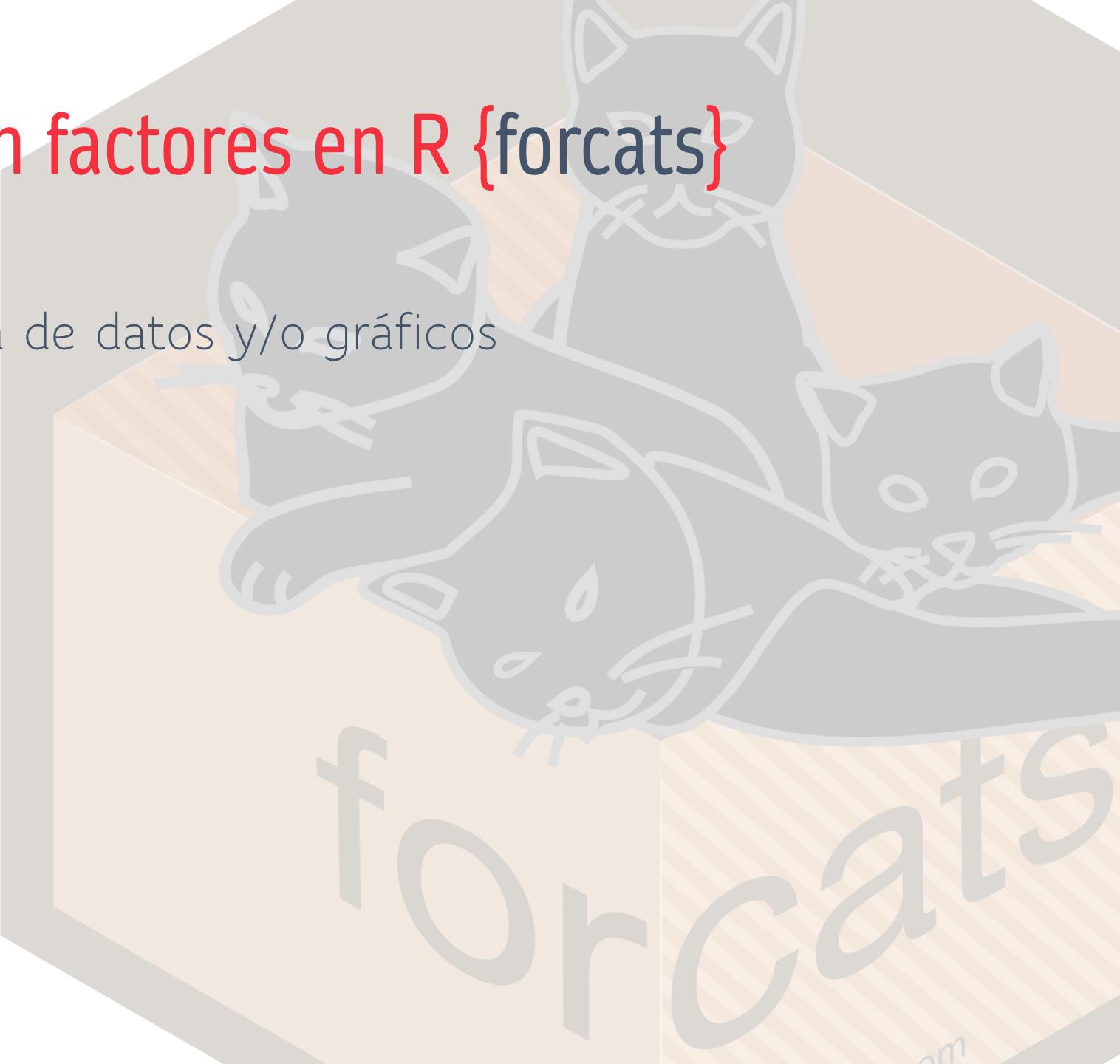
```
##
```

nombre	ojos	pelo	ojos_f	pelo_f
Length:5	Length:5	Min. :1.0	azules:2	lacio :2
Class :character	Class :character	1st Qu.:1.0	negros:2	rizado:2
Mode :character	Mode :character	Median :1.5	verdes:1	NA's :1
		Mean :1.5	grises:0	
		3rd Qu.:2.0		
		Max. :2.0		
		NA's :1		



Trabajando con factores en R {forcats}

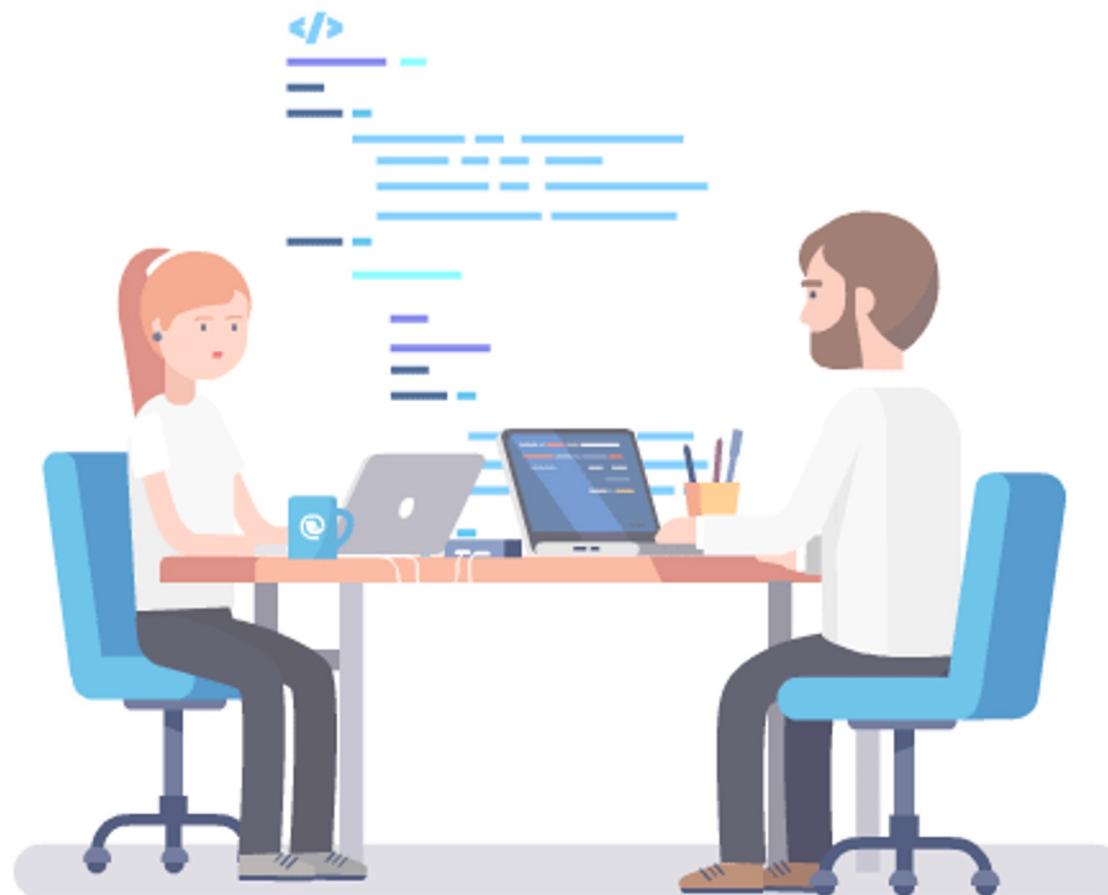
- Reordenar niveles en malla de datos y/o gráficos
- Rectificar niveles
- Agregar niveles





Su turno...

- Importe la malla **promedios_2021_ps.csv**
- Nombre a su objeto como **contam**
- Columnas en formato **snake case**





```
contam <-  
  read_csv("./data/promedios_2021_ps.csv", skip = 8) %>%  
  clean_names() %>%  
  print()
```

Rows: 6692 Columns: 5

— Column specification —

Delimiter: ","
chr (3): date, id_station, id_parameter
dbl (2): value, unit

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

```
# A tibble: 6,692 × 5  
  date      id_station id_parameter value  unit  
  <chr>     <chr>       <chr>        <dbl> <dbl>  
1 01/01/2021 ACO        PM10         70    2  
2 01/01/2021 ATI        PM10         44    2  
3 01/01/2021 CUT        PM10         73    2  
4 01/01/2021 FAC        PM10         43    2  
5 01/01/2021 MER        PM10         56    2  
6 01/01/2021 MER        PM2.5        40    2  
7 01/01/2021 NEZ        PM2.5        29    2  
8 01/01/2021 PED        PM10         34    2  
9 01/01/2021 PED        PM2.5        25    2  
10 01/01/2021 SAG       PM10         59    2  
# ... with 6,682 more rows
```



Su turno...

summary(contam)

date	id_station	id_parameter	value	unit
Length:6692	Length:6692	Length:6692	Min. : 2.00	Min. :2
Class : character	Class : character	Class : character	1st Qu.: 17.00	1st Qu.:2
Mode : character	Mode : character	Mode : character	Median : 25.00	Median :2
			Mean : 30.46	Mean :2
			3rd Qu.: 39.00	3rd Qu.:2
			Max. :158.00	Max. :2

- Identifique en la malla **contam** las variables que requieran codificarse como factor
- Convierta a factor las variables identificadas.
- Nombre a las nuevas columnas como **site** y **pollutant**.
- Ejecute el comando **summary**.
- Elimine las columnas **id_station**, **id_parameter** y **unit**.
- Renombre a la columna **value** como **concentration**.



```
contam <-
  contam %>%
  mutate(site = factor(id_station),
        pollutant = factor(id_parameter)) %>%
  print()
```

```
summary(contam)
```

	date	id_station	id_parameter	value	unit	site	pollutant
Length:6692	Length:6692	Length:6692	Min. : 2.00	Min. :2	BJU : 482	PM10 :3529	
Class :character	Class :character	Class :character	1st Qu.: 17.00	1st Qu.:2	MER : 478	PM2.5:3163	
Mode :character	Mode :character	Mode :character	Median : 25.00	Median :2	PED : 470		
			Mean : 30.46	Mean :2	TLA : 432		
			3rd Qu.: 39.00	3rd Qu.:2	SFE : 386		
			Max. :158.00	Max. :2	CAM : 364		
					(Other):4080		

```
contam <-
  contam %>%
  select(-c(id_station, id_parameter, unit)) %>%
  rename(concentration = value)
  print()
```

```
summary(contam)
```

	date	concentration	site	pollutant
Length:6692	Min. : 2.00	BJU : 482	PM10 :3529	
Class :character	1st Qu.: 17.00	MER : 478	PM2.5:3163	
Mode :character	Median : 25.00	PED : 470		
	Mean : 30.46	TLA : 432		
	3rd Qu.: 39.00	SFE : 386		
	Max. :158.00	CAM : 364		
	(Other):4080			



Fechas con código base

Columna o vector con fechas en algún formato consistente

- mes-día-año
- año/mes/dia

Formato de la fecha



```
as.Date(fecha, format)
```

malla

fecha	valor
12/11/19	12.1
31/01/21	13.4
28/02/17	11.8
03/09/18	14.4
22/11/20	12.7

```
malla %>%  
  mutate(fecha = as.Date(fecha, format = "%d/%m/%y"))
```

fecha	valor
2019-11-12	12.1
2021-01-31	13.4
2017-02-28	11.8
2018-09-03	14.4
2020-11-22	12.7



Fechas con código base

malla

fecha	valor
12-11-2019	12.1
31-01-2021	13.4
28-02-2017	11.8
03-09-2018	14.4
22-11-2020	12.7

%d %m %Y

malla %>%

```
mutate(fecha = as.Date(fecha, format = "%d-%m-%Y"))
```

fecha	valor
2019-11-12	12.1
2021-01-31	13.4
2017-02-28	11.8
2018-09-03	14.4
2020-11-22	12.7

Code	Value
%d	Day of the month (decimal number)
%m	Month (decimal number)
%b	Month (abbreviated)
%B	Month (full name)
%y	Year (2 digit)
%Y	Year (4 digit)



Fechas usando el paquete {lubridate}





malla

fecha	valor
12-11-2019	12.1
31-01-2021	13.4
28-02-2017	11.8
03-09-2018	14.4
22-11-2020	12.7

fecha	valor
12-2019-11	12.1
31-2021-01	13.4
28-2017-02	11.8
03-2018-09	14.4
22-2020-11	12.7

```
malla %>%  
  mutate(fecha = dmy(fecha))
```

fecha	valor
2019-11-12	12.1
2021-01-31	13.4
2017-02-28	11.8
2018-09-03	14.4
2020-11-22	12.7

```
malla %>%  
  mutate(fecha = dym(fecha))
```





```
today()                      # devuelve la fecha actual  
  
year("2020-11-05")          # devuelve el año de la fecha indicada  
  
month("2020-11-05")         # devuelve el mes de la fecha indicada  
  
day("2020-11-05")           # devuelve el día de la fecha indicada  
  
wday("2020-11-05")          # devuelve el número de la semana de la fecha indicada  
  
yday("2020-11-05")          # devuelve el número de día del año de la fecha indicada
```

```
malla %>%  
  mutate(mes = month(fecha, label = TRUE))
```

fecha	valor	mes
2019-11-12	12.1	nov
2021-01-31	13.4	ene
2017-02-28	11.8	feb



Dates and times with lubridate :: CHEAT SHEET

<https://lubridate.tidyverse.org>



Date-times



2017-11-28 12:00:00

2017-11-28 12:00:00

A **date-time** is a point on the timeline, stored as the number of seconds since 1970-01-01 00:00:00 UTC
`dt <- as_datetime(1511870400)
"2017-11-28 12:00:00 UTC"`

PARSE DATE-TIMES (Convert strings or numbers to date-times)

- Identify the order of the year (**y**), month (**m**), day (**d**), hour (**h**), minute (**m**) and second (**s**) elements in your data.
- Use the function below whose name replicates the order. Each accepts a wide variety of input formats.

2017-11-28T14:02:00

ymd_hms(), ymd_hm(), ymd_h().
ymd_hms("2017-11-28T14:02:00")

2017-22-12 10:00:00

ymd_hms(), ydm_hm(), ydm_h().
ydm_hms("2017-22-12 10:00:00")

11/28/2017 1:02:03

mdy_hms(), mdy_hm(), mdy_h().
mdy_hms("11/28/2017 1:02:03")

1 Jan 2017 23:59:59

dmy_hms(), dmy_hm(), dmy_h().
dmy_hms("1 Jan 2017 23:59:59")

20170131

ymd(), ydm(). ymd(20170131)

July 4th, 2000

mdy(), myd(). mdy("July 4th, 2000")

4th of July '99

dmy(), dyd(). dmy("4th of July '99")

2001: Q3

yq() Q for quarter. yq("2001: Q3")

2:01

hms::hms() Also lubridate::hms(), hm() and ms(), which return periods.* hms::hms(sec = 0, min = 1, hours = 2)

2017.5

date_decimal(decimal, tz = "UTC")
date_decimal(2017.5)



2017-11-28

A **date** is a day stored as the number of days since 1970-01-01
`d <- as_date(17498)
"2017-11-28"`

12:00:00

An **hms** is a **time** stored as the number of seconds since 00:00:00
`t <- hms::as.hms(85)
00:01:25`

GET AND SET COMPONENTS

Use an accessor function to get a component. Assign into an accessor function to change a component in place.

```
d ## "2017-11-28"  
day(d) ## 28  
day(d) <- 1  
d ## "2017-11-01"
```

2018-01-31 11:59:59

date(x) Date component. **date(dt)**

2018-01-31 11:59:59

year(x) Year. **year(dt)**
isoyear(x) The ISO 8601 year.
epiyear(x) Epidemiological year.

2018-01-31 11:59:59

month(x, label, abbr) Month.
month(dt)

2018-01-31 11:59:59

day(x) Day of month. **day(dt)**
wday(x, label, abbr) Day of week.
qday(x) Day of quarter.

2018-01-31 11:59:59

hour(x) Hour. **hour(dt)**

2018-01-31 11:59:59

minute(x) Minutes. **minute(dt)**

2018-01-31 11:59:59

second(x) Seconds. **second(dt)**

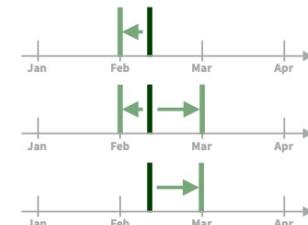
X

X

X



Round Date-times



floor_date(x, unit = "second")
Round down to nearest unit.
floor_date(dt, unit = "month")

round_date(x, unit = "second")
Round to nearest unit.
round_date(dt, unit = "month")

ceiling_date(x, unit = "second", change_on_boundary = NULL)
Round up to nearest unit.
ceiling_date(dt, unit = "month")

rollback(dates, roll_to_first = FALSE, preserve_hms = TRUE)
Roll back to last day of previous month. **rollback**(dt)

Stamp Date-times

stamp() Derive a template from an example string and return a new function that will apply the template to date-times. Also **stamp_date()** and **stamp_time()**.

1. Derive a template, create a function
`sf <- stamp("Created Sunday, Jan 17, 1999 3:34")`

Tip: use a date with day > 12

2. Apply the template to dates
`sf(ymd("2010-04-05"))
[1] "Created Monday, Apr 05, 2010 00:00"`

Time Zones

R recognizes ~600 time zones. Each encodes the time zone, Daylight Savings Time, and historical calendar variations for an area. R assigns one time zone per vector.

Use the **UTC** time zone to avoid Daylight Savings.

OlsonNames() Returns a list of valid time zone names. **OlsonNames()**



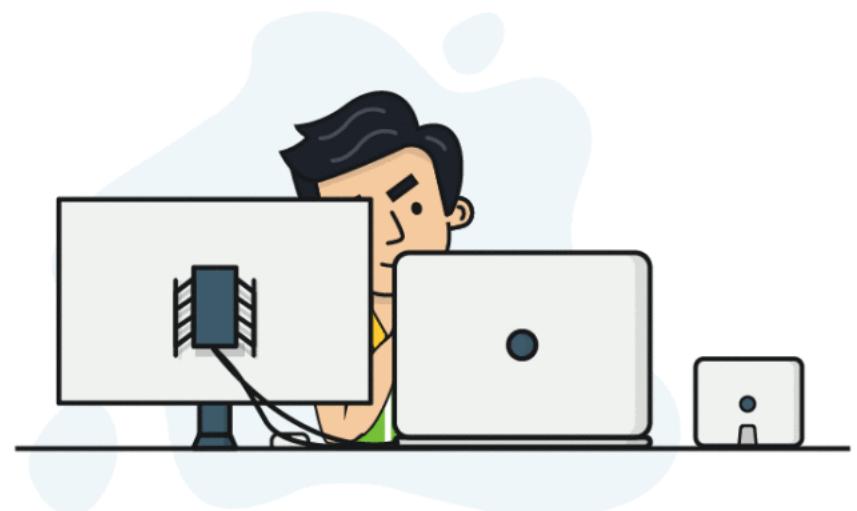
with_tz(time, timezone = "") Get the same date-time in a new time zone (a new clock time). **with_tz**(dt, "US/Pacific")

force_tz(time, timezone = "") Get the same clock time in a new time zone (a new date-time). **force_tz**(dt, "US/Pacific")



Su turno...

- Convierta a formato de fecha la columna `date`
- Genere columnas `day`, `month` (con etiqueta), `year`
- Guarde el objeto con el mismo nombre
- Genere el `summary` de la malla de datos





```
contam <-  
  contam %>%  
  mutate(date = as.Date(date, format = "%d/%m/%Y"),  
         day = day(date),  
         month = month(date, label = TRUE),  
         year = year(date)) %>%  
  print()
```

A tibble: 6,692 × 7

	date	concentration	site	pollutant	day	month	year
	<date>	<dbl>	<fct>	<fct>	<int>	<ord>	<dbl>
1	2021-01-01	70	ACO	PM10	1	ene	2021
2	2021-01-01	44	ATI	PM10	1	ene	2021
3	2021-01-01	73	CUT	PM10	1	ene	2021
4	2021-01-01	43	FAC	PM10	1	ene	2021
5	2021-01-01	56	MER	PM10	1	ene	2021
6	2021-01-01	40	MER	PM2.5	1	ene	2021
7	2021-01-01	29	NEZ	PM2.5	1	ene	2021
8	2021-01-01	34	PED	PM10	1	ene	2021
9	2021-01-01	25	PED	PM2.5	1	ene	2021
10	2021-01-01	59	SAG	PM10	1	ene	2021

... with 6,682 more rows

```
summary(contam)
```

	date	concentration	site	pollutant	day	month	year
Min.	:2021-01-01	Min. : 2.00	BJU : 482	PM10 :3529	Min. : 1.00	jul : 919	Min. :2021
1st Qu.	:2021-03-06	1st Qu.: 17.00	MER : 478	PM2.5:3163	1st Qu.: 8.00	may : 917	1st Qu.:2021
Median	:2021-05-07	Median : 25.00	PED : 470		Median :16.00	jun : 851	Median :2021
Mean	:2021-05-04	Mean : 30.46	TLA : 432		Mean :15.69	mar : 843	Mean :2021
3rd Qu.	:2021-07-03	3rd Qu.: 39.00	SFE : 386		3rd Qu.:23.00	ene : 834	3rd Qu.:2021
Max.	:2021-08-31	Max. :158.00	CAM : 364		Max. :31.00	ago : 831	Max. :2021
			(Other):4080			(Other):1497	



Malla incial

Malla procesada

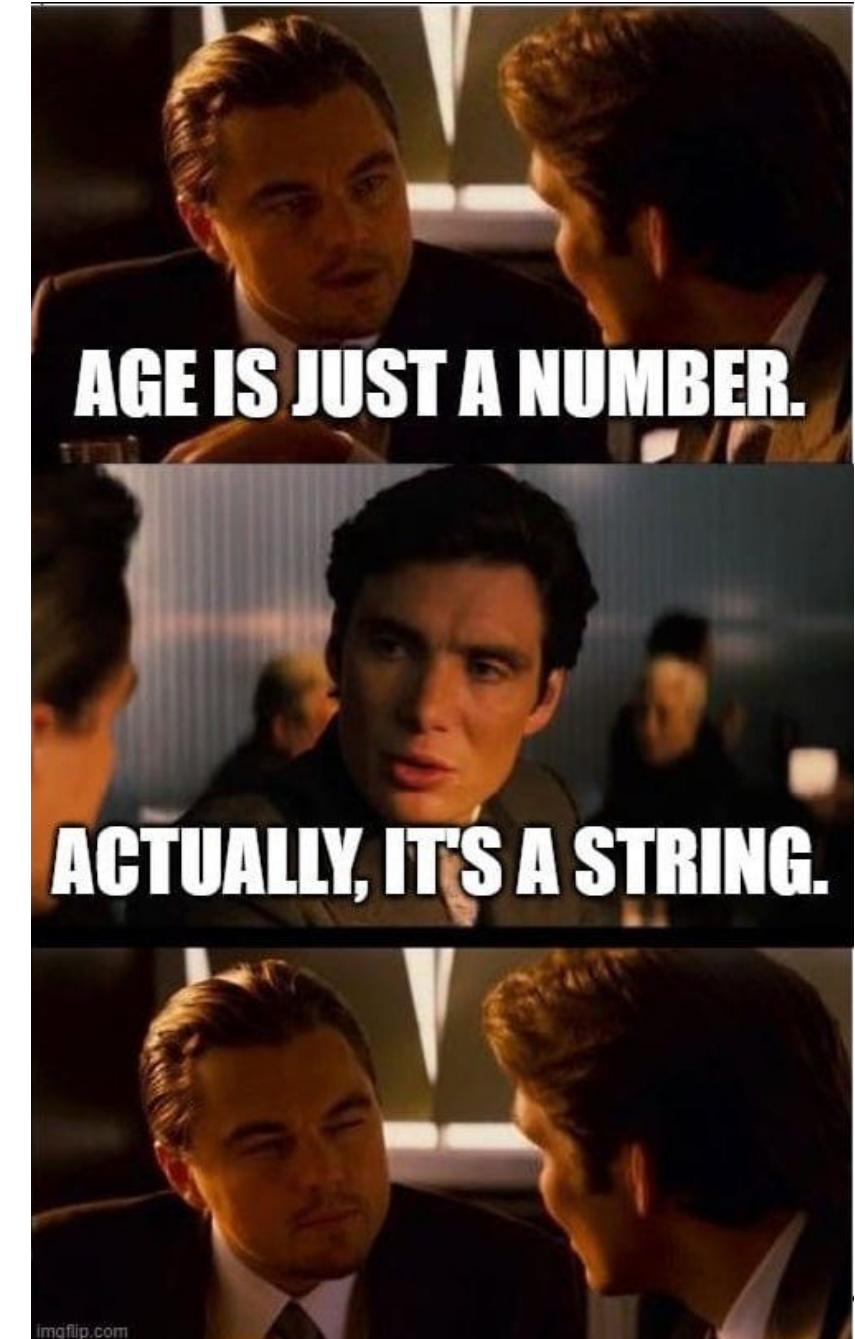
date	concentration	site	pollutant	day	month	year
Min. :2021-01-01	Min. : 2.00	BJU : 482	PM10 :3529	Min. : 1.00	jul : 919	Min. :2021
1st Qu.:2021-03-06	1st Qu.: 17.00	MER : 478	PM2.5:3163	1st Qu.: 8.00	may : 917	1st Qu.:2021
Median :2021-05-07	Median : 25.00	PED : 470		Median :16.00	jun : 851	Median :2021
Mean :2021-05-04	Mean : 30.46	TLA : 432		Mean :15.69	mar : 843	Mean :2021
3rd Qu.:2021-07-03	3rd Qu.: 39.00	SFE : 386		3rd Qu.:23.00	ene : 834	3rd Qu.:2021
Max. :2021-08-31	Max. :158.00	CAM : 364	(Other):4080	Max. :31.00	ago : 831	Max. :2021



Su turno...

- Filtre el contaminante **PM_{2.5}**
- Renombre la columna concentration como **pm25**
- Genere una nueva malla sólo con las columnas **date**, **site** y **pm25**, nombre al objeto como **pm25**
- Exporte su malla **pm25** con el nombre **pm25_cdmx_2021** con extensión **rds** y **csv**







str_c(..., sep = "")

str_c

Permite concatenar (unir) el texto de dos o más columnas en una sola.

```
dataset %>%  
  mutate(folio = str_c(posgrado, matricula, sexo)) %>%  
  print()
```

matricula	sexo	Posgrado	folio
0718	1	MCSA	MCSA07181
2183	2	MCEP	MCEP21832
1241	1	MCSS	MCSS12411
0213	2	MCSA	MCSA02132
4386	2	MCBI	MCBI43862



str_c(..., sep = "")

```
dataset %>%  
  mutate(folio = str_c(posgrado,  
                      matricula,  
                      sexo,  
                      sep = "-")) %>%  
  print()
```

sep = ""

Permite agregar un separador entre los elementos a concatenar (por ejemplo, un guión)

matricula	sexo	Posgrado	folio
0718	1	MCSA	MCSA-0718-1
2183	2	MCEP	MCEP-2183-2
1241	1	MCSS	MCSS-1241-1
0213	2	MCSA	MCSA-0213-2
4386	2	MCBI	MCBI-4386-2



1
001

str_pad(string, width, side = c("left", "right", "both"), pad = " ")

Vector o variable
a "inflar"

De qué lado se pone el texto que va
a "inflar" (por defecto a la izquierda)

Longitud que debe
tener el texto

Texto con el que
se va a "inflar"

```
malla %>%  
  mutate(clave_mun = str_pad(mun, 3, pad = "0"))
```

mun	municipio	clave_mun
1	San Juan	001
72	Santa María	072
348	Guadalupe	348
9	San Pablo	009



Concatenar c() y str_pad()

```
malla %>%  
  mutate(folio = str_c(id, nombre))
```

id	nombre	folio
1	Juan	1Juan
2	Pedro	2Pedro
3	Felipe	3Felipe

```
malla %>%  
  mutate(folio = str_c(id, nombre, sep = "-"))
```

id	nombre	folio
1	Juan	1-Juan
2	Pedro	2-Pedro
3	Felipe	3-Felipe

```
malla %>%  
  mutate(folio = str_c(str_pad(id, 2, pad = "0"),  
                      nombre,  
                      sep = "-"))
```

id	nombre	folio
1	Juan	01-Juan
2	Pedro	02-Pedro
3	Felipe	03-Felipe



str_sub(texto, start = 1, end = 4)

```
malla %>%  
  mutate(posgrado = str_sub(folio,  
                            start = 1,  
                            end = 4)) %>%  
  
  print()
```

matricula	sexo	folio	posgrado
0718	1	MCSA-0718-1	MCSA
2183	2	MCEP-2183-2	MCEP
1241	1	MCSS-1241-1	MCSS
0213	2	MCSA-0213-2	MCSA
4386	2	MCBI-4386-2	MCBI

str_sub

Permite extraer texto de un vector.

start =

En el ejemplo, el número 1 indica que se iniciará a extraer el texto a partir del primer carácter.

end =

En el ejemplo, el número 4 indica que se finalizará la extracción de texto en el cuarto carácter.



str_sub(texto, start = 5L)

```
malla %>%  
  mutate(matri_sexo = str_sub(folio,  
                             start = 6)) %>%  
  print()
```

matricula	sexo	folio	matri_sexo
0718	1	MCSA-0718-1	0718-1
2183	2	MCEP-2183-2	2183-2
1241	1	MCSS-1241-1	1241-1
0213	2	MCSA-0213-2	0213-2
4386	2	MCBI-4386-2	4386-2

str_sub

Permite extraer texto de un vector.

start =

En el ejemplo, el número 5 indica que se iniciará a extraer el texto a partir del quinto carácter.

end =

Si no se indica este argumento entonces el texto restante a partir del carácter de inicio es seleccionado.



```
sun <- read_csv("./data/Base_SUN_2018.csv") %>% clean_names() %>% print()
```

Rows: 1089 Columns: 9

— Column specification

Delimiter: ","

chr (8): CVE_ENT, NOM_ENT, CVE_MUN, NOM_MUN, CVE_LOC, NOM_LOC, CVE_SUN, NOM_SUN

dbl (1): POB_2018

i Use `spec()` to retrieve the full column specification for this data.

i Specify the column types or set `show_col_types = FALSE` to quiet this message.

```
# A tibble: 1,089 × 9
```

cve_ent	nom_ent	cve_mun	nom_mun	cve_loc	nom_loc	cve_sun	nom_sun	pob_2018
		<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<dbl>
1 01	Aguascalientes	01011	"San Francisco de ...	NA	NA	M01.01	Aguascalie...	42531
2 01	Aguascalientes	01005	"Jes\xfas Mar\xeda"	NA	NA	M01.01	Aguascalie...	116700
3 01	Aguascalientes	01001	"Aguascalientes"	NA	NA	M01.01	Aguascalie...	897331
4 02	Baja California	02005	"Playas de Rosarit...	NA	NA	M02.03	Tijuana	110683
5 02	Baja California	02003	"Tecate"	NA	NA	M02.03	Tijuana	115570
6 02	Baja California	02004	"Tijuana"	NA	NA	M02.03	Tijuana	1798741
7 02	Baja California	02002	"Mexicali"	NA	NA	M02.02	Mexicali	1065882
8 02	Baja California	02001	"Ensenada"	NA	NA	M02.01	Ensenada	542896
9 03	Baja California Sur	03003	"La Paz"	NA	NA	M03.01	La Paz	313204
10 04	Campeche	04002	"Campeche"	NA	NA	M04.01	Campeche	298741

... with 1,079 more rows



Averiguamos la codificación que mejor se adapte a nuestra malla de datos

```
guess_encoding("./ruta/file.csv", n_max = 1000)
```

```
# A tibble: 2 × 2
  encoding  confidence
  <chr>      <dbl>
1 ISO-8859-1    0.36
2 ISO-8859-2    0.26
```

Importamos la malla con la codificación recomendada

```
read_csv("./ruta/file.csv", locale = readr::locale(encoding = "ISO-8859-1"))
```

```
# A tibble: 1,089 × 9
  cve_ent nom_ent          cve_mun nom_mun          cve_loc nom_loc cve_sun nom_sun      pob_2018
  <chr>   <chr>          <chr>   <chr>          <chr>   <chr>   <chr>   <chr>      <dbl>
1 01      Aguascalientes  01011   San Francisco de l... NA       NA       M01.01  Aguascalie...  42531
2 01      Aguascalientes  01005   Jesús María        NA       NA       M01.01  Aguascalie...  116700
3 01      Aguascalientes  01001   Aguascalientes      NA       NA       M01.01  Aguascalie...  897331
4 02      Baja California 02005   Playas de Rosarito NA       NA       M02.03  Tijuana     110683
5 02      Baja California 02003   Tecate            NA       NA       M02.03  Tijuana     115570
```



Generar variables condicionales case_when()

mall

nombre	edad
Luisa	27
Juana	31
Petra	28
Maria	41
Andrea	33

```
malla %>%
  mutate(edad_gpo = case_when(edad >= 20 & edad < 30 ~ "20 a 29",
                               edad >= 30 & edad < 40 ~ "30 a 39",
                               edad >= 40 ~ "40 y más")) %>%
  print()
```

nombre	edad	edad_gpo
Luisa	27	"20 a 29"
Juana	31	"30 a 39"
Petra	28	"20 a 29"
Maria	41	"40 y más"
Andrea	33	"30 a 39"



Generar variables condicionales case_when()

malla

nombre	edad
Luisa	27
Juana	31
Petra	28
Maria	41
Andrea	33

```
malla %>%
  mutate(edad_gpo = case_when(edad >= 20 & edad < 30 ~ 2L,
                               edad >= 30 & edad < 40 ~ 3L,
                               edad >= 40 ~ 4L)) %>%
  print()
```

nombre	edad	edad_gpo
Luisa	27	2
Juana	31	3
Petra	28	2
Maria	41	4
Andrea	33	3

Generar variables condicionales case_when()

Generar nueva columna

Columna con los valores a identificar

- Condiciones
- mayor que
 - menor que
 - igual a

```
malla %>%
  mutate(new_var = case_when(variable == "condición_1" ~ "resultado",
                             variable == "condición_2" ~ "resultado",
                             TRUE ~ variable))
```

- Valor de la condición
- número
 - texto
 - lógico

Valor a almacenar en la nueva variable

<code>x < y</code>	menor que
<code>x > y</code>	mayor que
<code>x == y</code>	igual a
<code>x <= y</code>	menor o igual a
<code>x >= y</code>	mayor o igual a
<code>x != y</code>	diferente de
<code>x %in% y</code>	perteneces a
<code>is.na(x)</code>	is NA
<code>!is.na(x)</code>	Distinto de NA



Recodificar valores

malla

nombre	sexo	edad	peso
Kevin	1	17	-99
Brayan	1	-99	61.7
Kimberly	98	15	51.9
Britany	2	16	59.3
Brandon	99	17	-88
Melany	2	-88	61.6

Genero columna, si ya existe entonces la sobreescribe.

Activar paquete en la sesión

Comando para convertir valores a NA

Columna que contiene los valores NA

```
library(fauxnaif)
malla %>%
  mutate(sexo = na_if_in(sexo, 98, 99),
        edad = na_if_in(edad, -88, -99),
        peso = na_if_in(peso, -88, -99)) %>%
  print()
```

nombre	sexo	edad	peso
Kevin	1	17	NA
Brayan	1	NA	61.7
Kimberly	NA	15	51.9
Britany	2	16	59.3
Brandon	NA	17	NA
Melany	2	NA	61.6

Valores a convertir a NA



Recodificar valores

Es posible hacer todo el proceso para varias columnas a la vez



```
library(fauxnaif)
malla %>%
  mutate(across(c(sexo:peso), na_if_in, -88, -99, 98, 99)) %>%
  print()
```

malla

nombre	sexo	edad	peso
Kevin	1	17	-99
Brayan	1	-99	61.7
Kimberly	98	15	51.9
Britany	2	16	59.3
Brandon	99	17	-88
Melany	2	-88	61.6

nombre	sexo	edad	peso
Kevin	1	17	NA
Brayan	1	NA	61.7
Kimberly	NA	15	51.9
Britany	2	16	59.3
Brandon	NA	17	NA
Melany	2	NA	61.6



Trabajar con NA

nombre	edad	peso	pelo
juan	18	68.3	lacio
eva	NA	70.1	NA
luis	19	69.4	lacio
ana	20	NA	lacio
mario	20	73.5	chino
edith	19	65.2	NA
david	21	76.4	NA

```
malla %>% drop_na()
```

```
malla %>% na.omit()
```

nombre	edad	peso	pelo
juan	18	68.3	lacio
luis	19	69.4	lacio
mario	20	73.5	chino



Trabajar con NA

nombre	edad	peso	pelo
juan	18	68.3	lacio
eva	NA	70.1	NA
luis	19	69.4	lacio
ana	20	NA	lacio
mario	20	73.5	chino
edith	19	65.2	NA
david	21	76.4	NA

```
malla %>% drop_na(pelo)
```

nombre	edad	peso	pelo
juan	18	68.3	lacio
luis	19	69.4	lacio
ana	20	NA	lacio
mario	20	73.5	chino



Trabajar con NA

nombre	edad	peso	pelo
juan	18	68.3	lacio
eva	NA	70.1	NA
luis	19	69.4	lacio
ana	20	NA	lacio
mario	20	73.5	chino
edith	19	65.2	NA
david	21	76.4	NA

```
malla %>% drop_na(edad, peso)
```

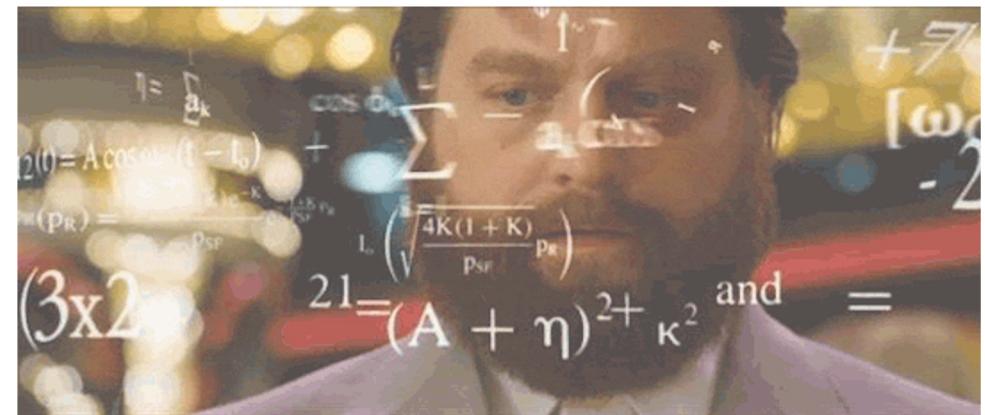
nombre	edad	peso	pelo
juan	18	68.3	lacio
luis	19	69.4	lacio
mario	20	73.5	chino
edith	19	65.2	NA
david	21	76.4	NA



Su turno...

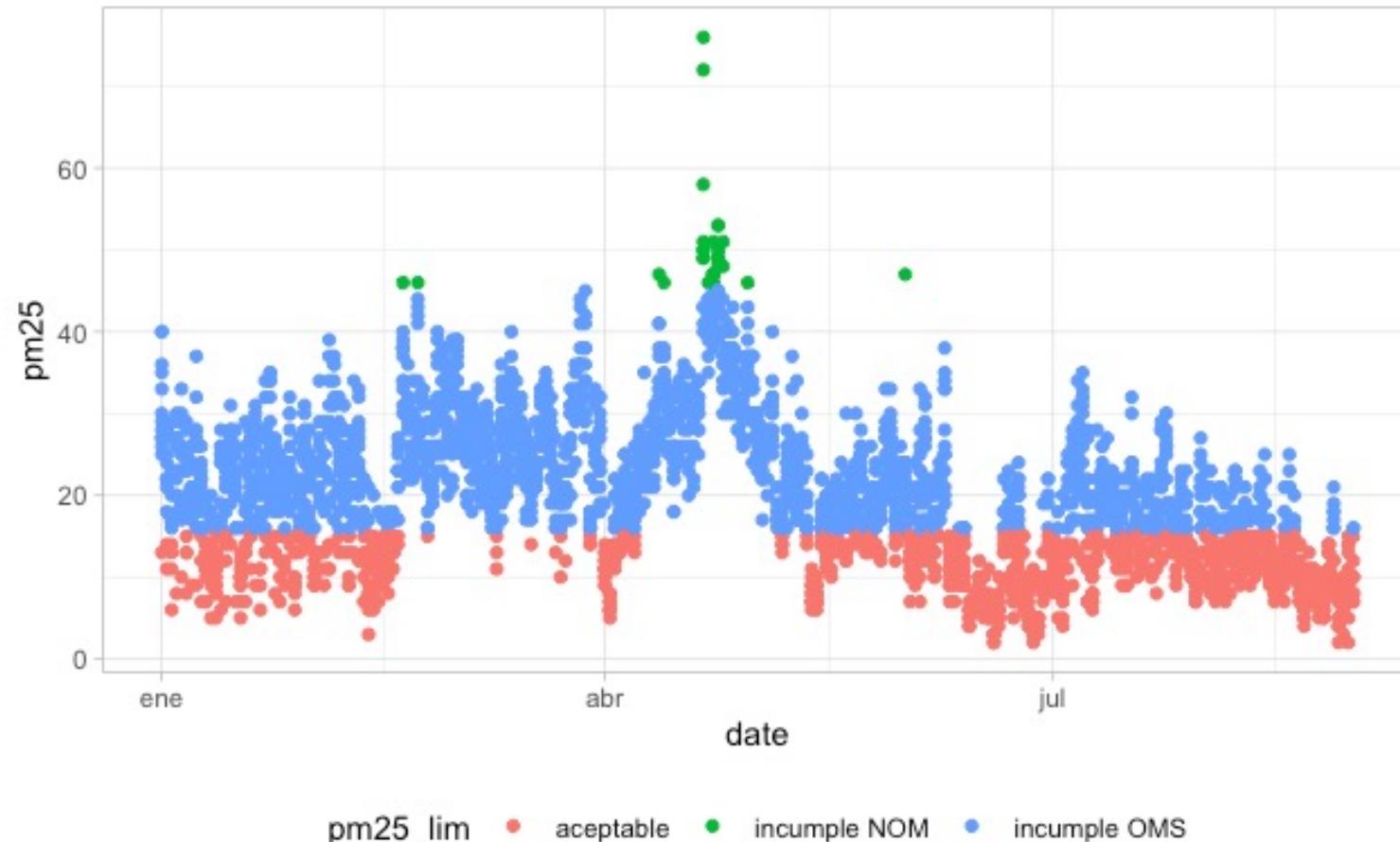
Realice las siguientes actividades a partir de la malla pm25

- Genere columna month de forma numérica {lubridate}
- Genere variable pm25_lim en la que clasifique la concentración de PM_{2.5} de acuerdo a si cumple la NOM o la guía OMS
 - Valor NOM ($\text{? } \mu\text{g}/\text{m}^3$): "incumple NOM"
 - Valor OMS ($\text{? } \mu\text{g}/\text{m}^3$): "incumple OMS"
 - Valor cumple valores límites: "Aceptable"
- Guarde el resultado en objeto de nombre pm25_dia



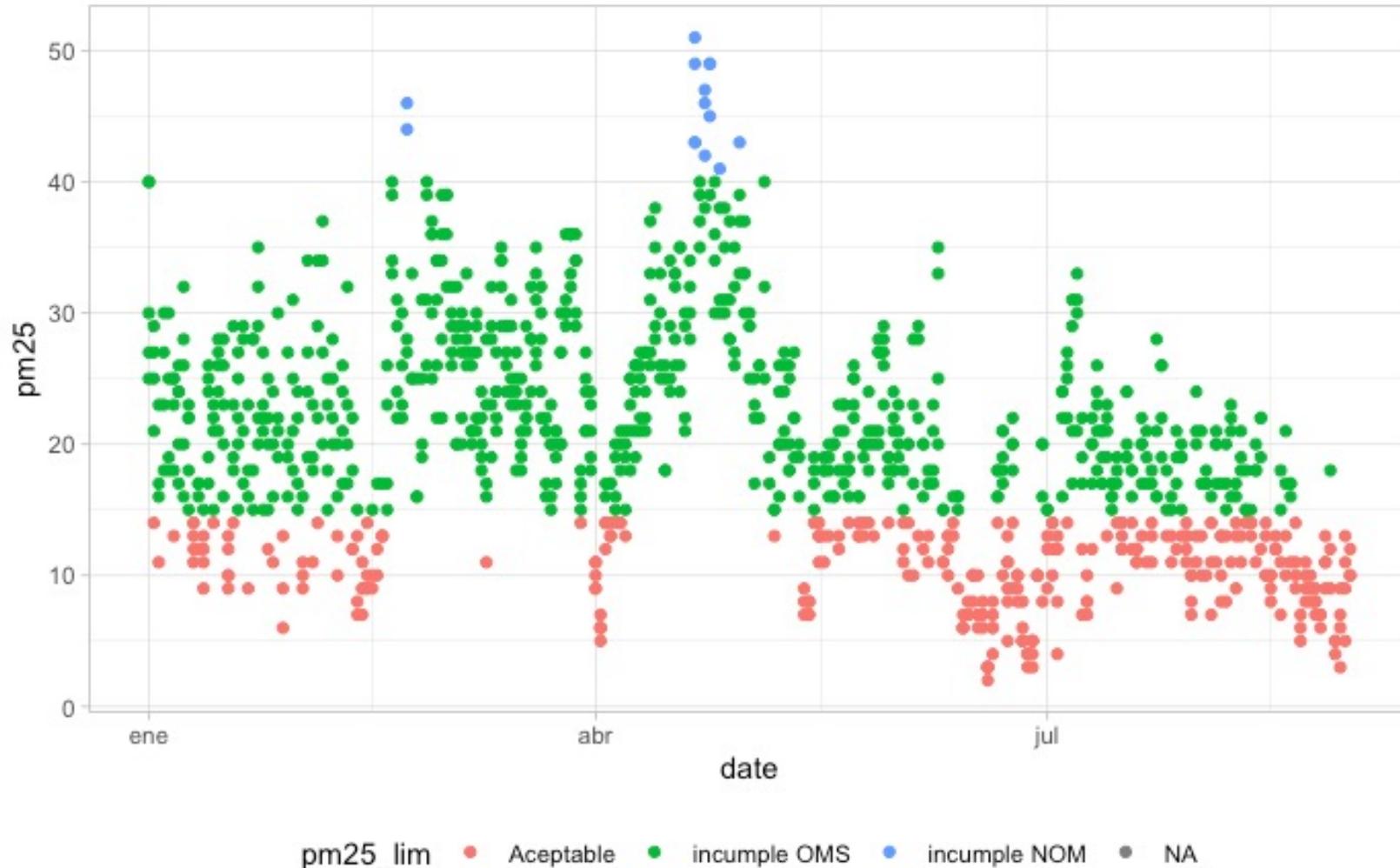


```
ggplot(data = pm25_dia) +  
  geom_point(aes(date, pm25, colour = pm25_lim)) +  
  theme_light()  
  theme(legend.position = "bottom")
```





```
ggplot(data = pm25_dia) +  
  geom_point(aes(date, pm25, colour = pm25_lim)) +  
  theme_light()  
  theme(legend.position = "bottom")
```





Editar componentes del gráfico

ggplot2 theme elements reference

Set minimal as the baseline theme:

```
theme_minimal() +  
  theme(element = element_type())
```

Use element_blank() to remove an element

Axis titles, text, ticks, and lines can be specified per axis using theme inheritance by putting .x/.y at the end of the theme element.

```
axis.line.y = element_line()
```

```
axis.title.y = element_text()
```

```
panel.grid.major = element_line()
```

```
panel.grid.minor = element_line()
```

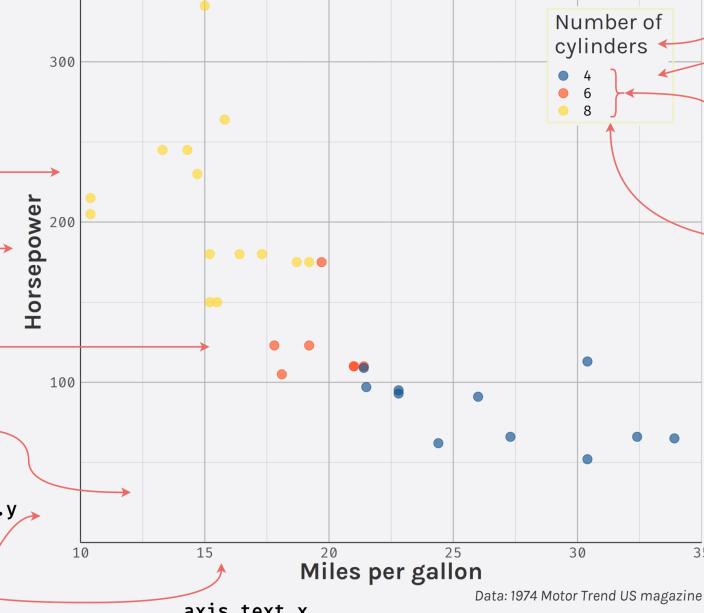
```
axis.text.y = element_text()
```

```
axis.text = element_text()
```

```
plot.title.position = "plot"  
plot.caption.position = "plot" } "plot" means that they will be aligned to the entire plot (instead of the panel)  
plot.title = element_text()  
plot.subtitle = element_text()
```

```
plot.margin = margin(25, 25, 25, 25)
```

Miles per Gallon & Horsepower
of 32 Automobiles (1973-74 models)



```
legend.title = element_text()
```

```
legend.background = element_rect()
```

```
legend.text = element_text()
```

```
legend.position = c(.85,.85) / "none" /  
  "left" / "right" /  
  "bottom" / "top"
```

```
plot.background = element_rect()
```

```
plot.caption = element_text()
```

isabella-b

`text = element_text() ←` modifications will be applied to all text elements

Full list of elements at ggplot2.tidyverse.org/reference/theme.html