

# Taller R – Rifa Valentina

## 2. Manipulación de variables de tiempo

26/01/2022


# Contenidos

- Manipulación avanzada de data frames (funciones `pivot` y combinación de data frames)
- Manipulación de fechas: paquete `lubridate`
- Visualización de datos con `ggplot2`
- Estimar desde diseños muestrales complejos (paquete `survey` y `srvyr`)
- Hacer funciones, procesos iterativos e introducción a paquetes en R



# Manipulacion de fechas

# Variables de tiempo en R

En este módulo **solo veremos fechas** , pero para horas, minutos y segundos la lógica es la misma.

Sin embargo, si bien es posible trabajar variables de tiempo sin herramientas dedicadas, sería **extremadamente engorroso**, y se requerirían herramientas medianamente sofisticadas para transformar estas variables en objetos con los que podamos operar.

Para eso R ofrece herramientas dedicadas especialmente a solucionarnos la vida. Podrían separarse en 2 tipos:

- Herramientas para organizar fechas en un formato reconocible.
- Herramientas que traduzcan estas fechas de formato estandarizado a números.

# Variables de tiempo en R

Y con números se pueden hacer muchas cosas: **operaciones matemáticas, gráficos**, etc. Mientras R por detrás trabaja con números, nosotros seguimos viendo sencillas y amigables fechas 🤖.

```
# Pueden reemplazar esta fecha por sus cumpleaños  
mi_cumple <- ("1993-09-30")  
str(mi_cumple)
```

```
## chr "1993-09-30"
```

```
mi_cumple <- as.Date(mi_cumple)  
str(mi_cumple)
```

```
## Date[1:1], format: "1993-09-30"
```

```
as.numeric(mi_cumple)
```

```
## [1] 8673
```

La función `as.Date()` nos dio un número. ¿Qué representa ese número?

# Variables de tiempo en R

```
as.numeric(as.Date("1970-01-01"))
```

```
## [1] 0
```

Es la distancia desde un momento **escogido de manera arbitraria**: el 1 de enero de 1970.

`as.Date()` es una función sencilla de usar, **pero no es muy robusta para el trabajo con fechas**.

```
mi_cumple <- as.Date("30-09-1993"); str(mi_cumple)
```

```
## Date[1:1], format: "0030-09-19"
```

No soluciona el problema del ordenamiento. Requiere asistentes para hacerlo.

# Variables de tiempo en R

Aún así, **no hay que descartarla**, es muy útil cuando el formato es la norma ISO 8661. Esta indica un formato YYYY-MM-DD y una cantidad de dígitos por parámetro (4-2-2).

Dentro del universo de `tidyverse` existe una **librería especializada para el tratamiento de fechas y horas**.

Se llama `lubridate` y su objetivo es hacer más intuitiva la manipulación y análisis de este tipo de variables.



# Variables de tiempo en R

Veamos algunos operadores básicos muy útiles.

R base tiene funciones para extraer la fecha y hora en el momento de la consulta.

```
Sys.Date() # La fecha de hoy
```

```
## [1] "2022-01-24"
```

```
Sys.time() # el momento exacto, con fecha, horas, minutos y segundos
```

```
## [1] "2022-01-24 13:17:23 -03"
```



# Usando lubridate

lubridate tiene funciones que hacen lo mismo, pero con un lenguaje más intuitivo.

```
library(lubridate) # cargamos lubridate
today()
```

```
## [1] "2022-01-24"
```

```
now()
```

```
## [1] "2022-01-24 13:17:23 -03"
```

Hay 2 formas principales para crear una fecha.

- Desde una cadena de caracteres o números.
- Desde componentes *date-time* individuales.

# Usando lubridate

## 1. Desde cadenas de caracteres

- La más habitual es a partir de cadenas de caracteres.
- Existen helpers en lubridate que automáticamente ordenan el formato de una variable fecha.
- Solo hay que ordenarlos de acuerdo al input.
- Se aceptan diferentes tipos de separadores.

```
ymd("1993-09-30")
```

```
## [1] "1993-09-30"
```

```
## No asimila bien el mes en espa  
mdy("Sep 30, 1993")
```

```
## [1] "1993-09-30"
```

```
dmy("30/sep/1993")
```

```
## [1] "1993-09-30"
```

# Usando lubridate

También se pueden crear fechas a partir de variables numéricas. Siempre y cuando respeten el orden y cantidad de dígitos.

```
ymd(20190322)
```

```
## [1] "2019-03-22"
```

```
dmy(22032019)
```

```
## [1] "2019-03-22"
```

# Usando lubridate

## 2. Creación desde componentes *date-time* individuales

A veces las fechas nos llegan en un *data frame* separadas en día, mes, año.

Debemos unir las para operarlas como objetos *date*.

Usaremos la base de nacimientos de EEVV 2017.

Usamos la función `make_date()` de `lubridate`.

```
# cargamos la base
library(readxl)
library(lubridate)

nac2017 <- read_excel("data/nac_2017.xlsx")
```

```
# seleccionamos día, mes, año de nacimiento y creamos una fecha
nac2017 %>%
  mutate(fecha_nac = make_date(year=ano_nac,
                                month=mes_nac,
                                day=dia_nac))
```

# Usando lubridate

```
## # A tibble: 5 x 4
##   dia_nac mes_nac ano_nac fecha_nac
##   <dbl>   <dbl>   <dbl> <date>
## 1     27     11    2017 2017-11-27
## 2     27      1    2017 2017-01-27
## 3     21      3    2017 2017-03-21
## 4     28      6    2017 2017-06-28
## 5     10      4    2017 2017-04-10
```

Si no hay variable día, utilizar carácter "01"

```
nac2017 %>%
  mutate(fecha_nac = make_date(year=ano_nac,
                                month=mes_nac,
                                day="01"))
```

# Mini ejercicio

Crear variable fecha de nacimiento y fecha de inscripción.

```
nac2017 <- nac2017 %>%  
  mutate(fecha_nac = make_date(ano_nac, mes_nac, dia_nac),  
         fecha_ins = make_date(ano_ins, mes_ins, dia_ins))
```

```
nac2017 %>% select(ano_nac, mes_nac, dia_nac, ano_ins, mes_ins,  
                  dia_ins, fecha_nac, fecha_ins) %>%  
  head()
```

```
## # A tibble: 6 x 8  
##   ano_nac mes_nac dia_nac ano_ins mes_ins dia_ins fecha_nac fecha_ins  
##   <dbl>   <dbl>   <dbl> <dbl>   <dbl>   <dbl> <date>   <date>  
## 1    2017     11     27    2017     11     30 2017-11-27 2017-11-30  
## 2    2017      1     27    2017      2      2 2017-01-27 2017-02-02  
## 3    2017      3     21    2017      3     23 2017-03-21 2017-03-23  
## 4    2017      6     28    2017      7      3 2017-06-28 2017-07-03  
## 5    2017      4     10    2017      4     13 2017-04-10 2017-04-13  
## 6    2017     10     14    2017     10     16 2017-10-14 2017-10-16
```

# Usando lubridate

Así como podemos componer una fecha, también podemos descomponerla.

```
mi_cumple <- dmy("30-09-1993")  
year(mi_cumple)
```

```
## [1] 1993
```

```
month(mi_cumple, label = T) # con label se pide la etiqueta
```

```
## [1] sept
```

```
## 12 Levels: ene < feb < mar < abr < may < jun < jul < ago < sept < ... < dic
```

```
mday(mi_cumple)
```

```
## [1] 30
```

```
wday(mi_cumple, label = T) # considera que el día 1 es el domingo
```

```
## [1] jue\\.
```

```
## Levels: dom\\. < lun\\. < mar\\. < mié\\. < jue\\. < vie\\. < sáb\\. 15/29
```

# Operaciones aritméticas

Por ejemplo, pueden saber cuántos días de vida tienen.

```
today() - ymd("1993-09-30")
```

```
## Time difference of 10343 days
```

Existe un **set de funciones** que sirven para operar sobre periodos de tiempo de una manera intuitiva y versatil: se llaman `periods` y algunos de ellos son:

```
days(1)
weeks(1)
months(1) # esta función es de R base
years(1)
```

¿Qué podemos hacer con ellos?

```
# ¿que fecha es en un año y un mes más?
today() + years(1) + months(1)
```

```
## [1] "2023-02-24"
```



# Operaciones aritméticas

Podemos, por ejemplo, crear una variable *deadline* que indique cuándo es un mes después de un punto inicial.

```
inicio <- as.Date("2020-08-30")  
inicio + months(1)
```

```
## [1] "2020-09-30"
```

Pero no es una función tan robusta. ¿Qué pasa con los meses de 31 días?

```
inicio <- ymd("2020-08-31") # esta otra función es parecida a as.Date  
inicio + months(1)
```

```
## [1] NA
```

No sabe qué hacer y entrega un NA. Pero `lubridate()` contiene operadores robustos para solucionarlo.

```
inicio %m+% months(1)
```

# Operaciones aritméticas

%m+% también funciona con años y días. También existe %m-% para restar periodos.

```
bisiesto <- ymd("2020-02-29")  
bisiesto %m+% years(1)
```

```
## [1] "2021-02-28"
```

```
bisiesto %m+% days(1)
```

```
## [1] "2020-03-01"
```

Además se pueden generar automáticamente varios periodos.

```
inicio <- ymd("2020-08-31")  
inicio %m+% months(1:6)
```

```
## [1] "2020-09-30" "2020-10-31" "2020-11-30" "2020-12-31" "2021-01-31"  
## [6] "2021-02-28"
```

# Operaciones aritméticas

También podemos calcular **intervalos de tiempo** entre dos momentos de manera consistente.

Para eso utilizamos el operador `%--%`.

```
siguiente_año <- today() + years(1)
(today() %--% siguiente_año) / days(1) # diferencia en días
```

```
## [1] 365
```

Para encontrar cuántos períodos caen dentro de un intervalo, con `%/%` pueden obtener la división entera:

```
(today() %--% siguiente_año) / weeks(1)
```

```
## [1] 52.14286
```

Ahora con `%/%`.

```
(today() %--% siguiente_año) %/% weeks(1)
```

# Mini-ejercicio

1- Generar "dif\_days" entre fecha nacimiento e inscripción

```
nac2017 <- nac2017 %>%  
  mutate(dif_days = (fecha_nac %--% fecha_ins) / days(1))
```

2- Generar "dif\_weeks".

```
nac2017 <- nac2017 %>%  
  mutate(dif_weeks = (fecha_nac %--% fecha_ins) %/% weeks(1))
```

3- Generación tabla de resumen de estadísticos.

```
nac2017 %>% summarise(min_dif = min(dif_days),  
                      max_dif = max(dif_days),  
                      media_dif = mean(dif_days),  
                      median_dif = median(dif_days))
```

```
## # A tibble: 1 x 4  
##   min_dif max_dif media_dif median_dif  
##   <dbl>   <dbl>   <dbl>   <dbl>
```

# Usando lubridate

El uso de `lubridate` puede generar **cierta dificultad** en un principio.

Esto debido a la cantidad de operadores nuevos que ofrece (`%--%`, `%m+%`, `%m-%`, etc.).

Pero si trabajamos habitualmente con fechas u horas, y son un aspecto importante de nuestro trabajo, vale mucho la pena estudiarlos bien.

Pues `lubridate` ofrece herramientas precisas y robustas para el trabajo con datos temporales.

Que además son absolutamente compatibles con las librerías de `tidyverse`.

Pueden encontrar muchísima más información [aquí](#).

# Uso práctico

Para crear series de tiempo

```
# Identificar fecha inicio y fin  
inicio <- "2016-11-01"  
fin <- "2017-06-01"
```

```
# Contar meses entre medio  
meses <- ((ymd(inicio) %--% ymd(fin)) / months(1) )  
meses
```

```
## [1] 7
```

```
## Crear serie  
ano_mes_dia <- ymd(inicio) %m+% months(0:meses) %>% as.character()  
ano_mes_dia
```

```
## [1] "2016-11-01" "2016-12-01" "2017-01-01" "2017-02-01" "2017-03-01"  
## [6] "2017-04-01" "2017-05-01" "2017-06-01"
```

```
substr(ano_mes_dia,1,7) ## Si el dia molesta
```

# Uso práctico

Recomendación: si haces gráfico con fechas aprovecha y utiliza *lubridate*

Cuando visualizamos variables en **años** no hay mayor problema, pueden ser números. El tema empieza cuando hay meses o días.

Veamos series de empleo (ENE): tasa de desocupación

```
# link
url <- "https://www.ine.cl/docs/default-source/ocupacion-y-desocupacion,

# donde guardar
destfile <- "data/ine_desocupacion.xlsx"

# descargar
download.file(url, destfile, method = "curl")
```

# Uso práctico

```
# cargar la data
desocup <- readxl::read_excel("data/ine_desocupacion.xlsx", skip = 6, sheet = "Desocupados")
head(desocup, 4)
```

```
## # A tibble: 4 x 28
##   ...1    ...2      nota...3 `en miles...4` nota...5 `en miles...6` nota...7
##   <chr> <chr>      <lgl>          <dbl> <lgl>          <dbl> <lgl>
## 1 2010   Ene - Mar NA          13218. NA          7884. NA
## 2 2010   Feb - Abr NA          13236. NA          7897. NA
## 3 2010   Mar - May NA          13253. NA          7900. NA
## 4 2010   Abr - Jun NA          13270. NA          7906. NA
## # ... with 21 more variables: en miles...8 <dbl>, nota...9 <lgl>,
## #   en miles...10 <dbl>, nota...11 <lgl>, en miles...12 <dbl>, nota...13 <lgl>,
## #   en miles...14 <dbl>, nota...15 <lgl>, en miles...16 <dbl>, nota...17 <lgl>,
## #   en miles...18 <dbl>, nota...19 <lgl>, en miles...20 <dbl>, nota...21 <lgl>,
## #   en miles...22 <dbl>, nota...23 <lgl>, tasa (%)...24 <dbl>, nota...25 <lgl>,
## #   tasa (%)...26 <dbl>, nota...27 <lgl>, tasa (%)...28 <dbl>
```

¿Que se ve?

Fechas como carácter y año separado de meses. Además, meses como trimestres.



# Uso práctico

```
# Limpiar y renombrar
desocup <- desocup %>% janitor::clean_names() %>%
  select(x1,x2,tasa_percent_24) %>%
  filter(!is.na(x2)) %>%
  rename(ano=x1,mes=x2)
head(desocup,2)
```

```
## # A tibble: 2 x 3
##   ano   mes      tasa_percent_24
##   <chr> <chr>          <dbl>
## 1 2010 Ene - Mar        9.23
## 2 2010 Feb - Abr        8.84
```

Año ok, pero mes en español y trimestre.

```
desocup$mes <- tolower(substr(desocup$mes,1,3))
table(desocup$mes)
```

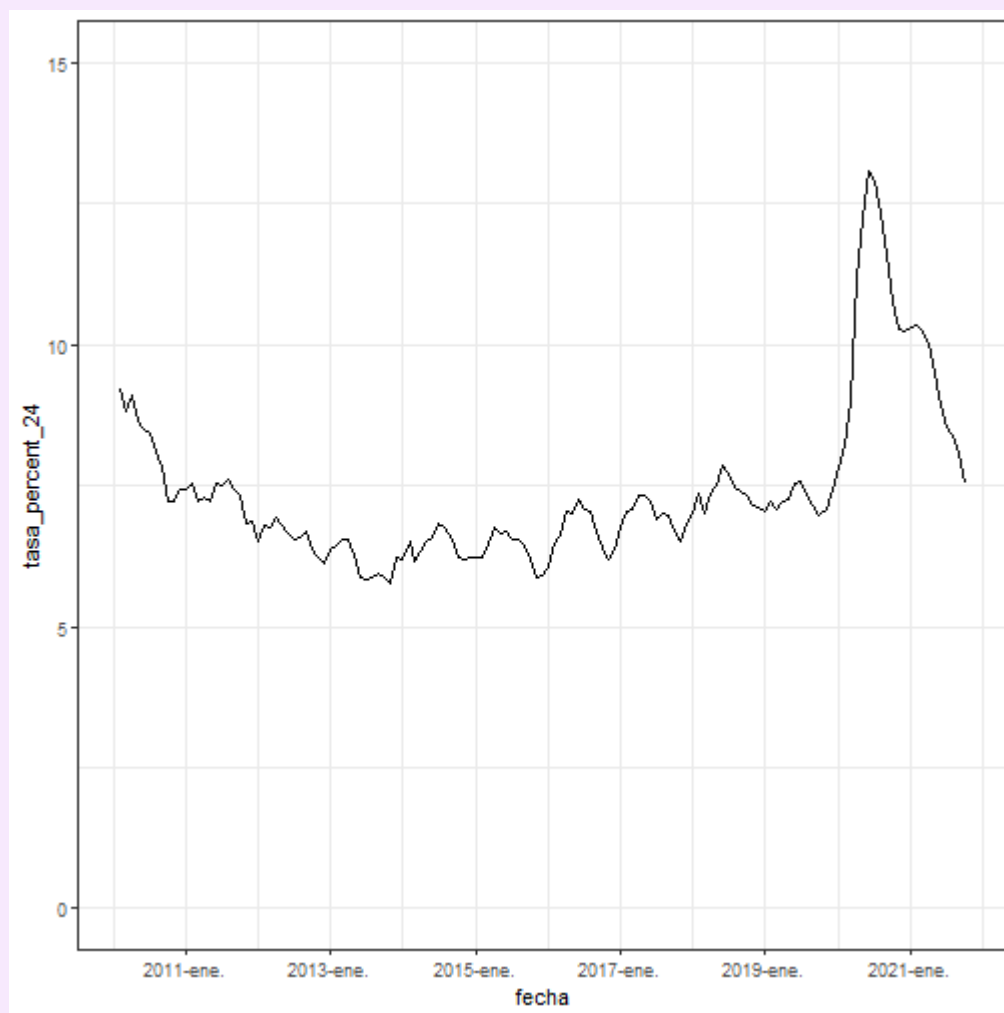
```
##
## abr ago dic ene feb jul jun mar may nov oct sep
##  12  12  11  12  12  12  12  12  12  11  11  12
```

# Uso práctico

```
desocup <- desocup %>%
  mutate(mes=case_when(mes == "ene" ~ 2,
                        mes == "feb" ~ 3,
                        mes == "mar" ~ 4,
                        mes == "abr" ~ 5,
                        mes == "may" ~ 6,
                        mes == "jun" ~ 7,
                        mes == "jul" ~ 8,
                        mes == "ago" ~ 9,
                        mes == "sep" ~ 10,
                        mes == "oct" ~ 11,
                        mes == "nov" ~ 12,
                        mes == "dic" ~ 1)) %>%
  mutate(fecha=make_date(year=ano,
                          month=mes))
```

```
## # A tibble: 3 x 4
##   ano    mes tasa_percent_24 fecha
##   <chr> <dbl>         <dbl> <date>
## 1 2010     2         9.23 2010-02-01
## 2 2010     3         8.84 2010-03-01
```

# Uso práctico



# El código

```
library(scales)
library(ggplot2)
desocup %>%
  ggplot(aes(x=fecha,y=tasa_percent_24))+
  geom_line()+
  theme_bw()+
  scale_x_date(labels = date_format("%Y-%b"),breaks='2 year' ) +
  scale_y_continuous(limits = c(0,15))
```

# Recursos utilizados

Ilustraciones de Allison Horst

Tutorial de `lubridate`

Capacitación INE variables de tiempo en R