Fechas

Jorge Loría

Nov 1, 2017

Trabajar con fechas siempre es un dolor

- ¿Todos los años tienen 365 días?

Trabajar con fechas siempre es un dolor

- ¿Todos los años tienen 365 días?
- ¿Todos los días tienen 24 horas?

Trabajar con fechas siempre es un dolor

- ¿Todos los años tienen 365 días?
- ¿Todos los días tienen 24 horas?
- ¿Todos los minutos tienen 60 segundos?

lubridate

La idea de este paquete es que sea lo más sencillo posible tratar con fechas y momentos del día.

```
library(dplyr)
library(lubridate)
today()
```

```
## [1] "2018-11-04"
```

```
now()
```

```
## [1] "2018-11-04 17:32:29 CST"
```

Formas de conseguir una fecha

- De texto: `'2018-11-05'`

Formas de conseguir una fecha

- De texto: `'2018-11-05'`
- Componentes individuales: Año $= 2018$, Mes $= 11$, Día $= 5$

Formas de conseguir una fecha

- De texto: `'2018-11-05'`
- Componentes individuales: Año $= 2018$, Mes $= 11$, Día $= 5$
- De otro objeto de fecha

De texto

De texto

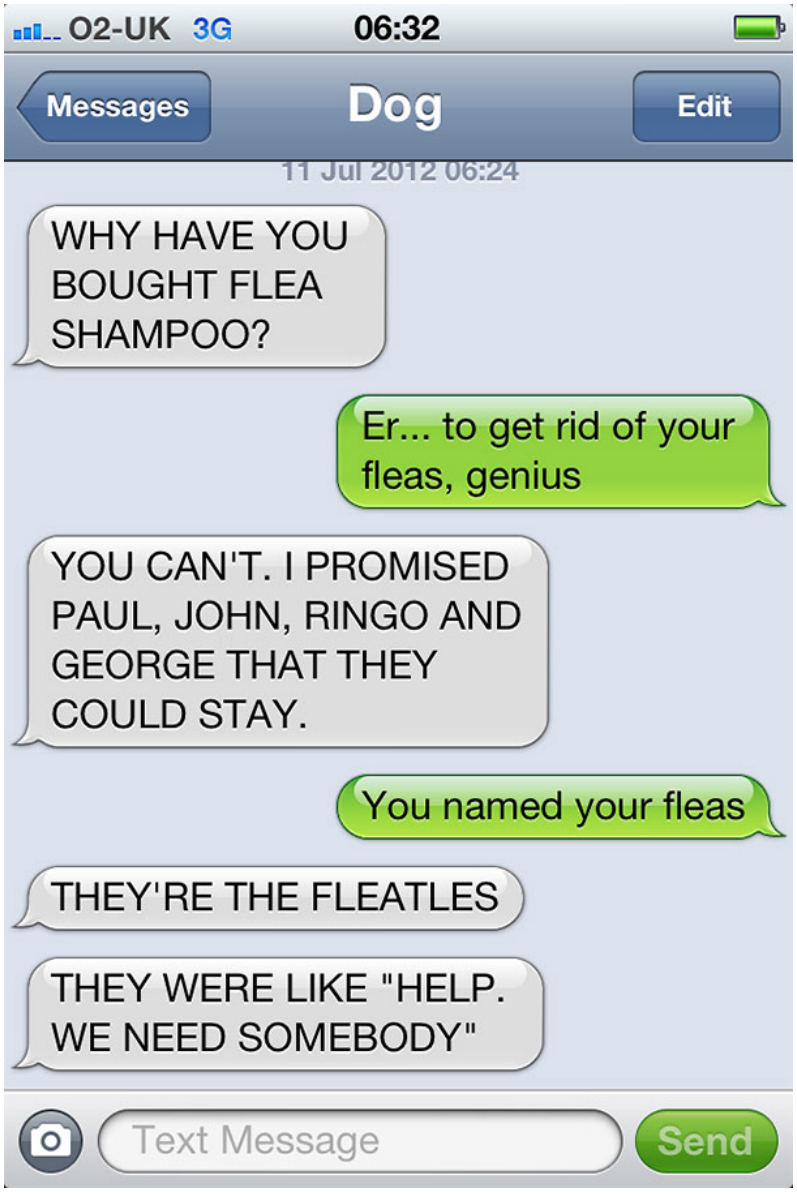


Figure 1: Pero no mensajes

De texto

De texto

Como hay muchas formas de escribir una misma fecha, para cada una de estas hay una función de `lubridate`, para interpretarla. Solo que estamos limitados a **inglés**. Pero todas las siguientes fechas representan la misma fecha:

```
fecha1 <- '2015-22-03'
fecha2 <- '2015-03-22'
fecha3 <- '3-22-2015'
fecha4 <- 'Mar-22-2015'
fecha5 <- 'March 22nd, 2015'
fecha6 <- '22/03/2015'
fecha7 <- 20150322
```

De texto

Como hay muchas formas de escribir una misma fecha, para cada una de estas hay una función de `lubridate`, para interpretarla. Solo que estamos limitados a **inglés**. Pero todas las siguientes fechas representan la misma fecha:

```
fecha1 <- '2015-22-03'
fecha2 <- '2015-03-22'
fecha3 <- '3-22-2015'
fecha4 <- 'Mar-22-2015'
fecha5 <- 'March 22nd, 2015'
fecha6 <- '22/03/2015'
fecha7 <- 20150322
```

Para convertirlas a un objeto tipo fecha usamos, respectivamente, las funciones: `ydm`, `ymd`, `mdy`, `mdy`, `mdy`, `dmy`, `ymd`.

De texto

Como hay muchas formas de escribir una misma fecha, para cada una de estas hay una función de `lubridate`, para interpretarla. Solo que estamos limitados a **inglés**. Pero todas las siguientes fechas representan la misma fecha:

```
fecha1 <- '2015-22-03'
fecha2 <- '2015-03-22'
fecha3 <- '3-22-2015'
fecha4 <- 'Mar-22-2015'
fecha5 <- 'March 22nd, 2015'
fecha6 <- '22/03/2015'
fecha7 <- 20150322
```

Para convertirlas a un objeto tipo fecha usamos, respectivamente, las funciones: `ydm`, `ymd`, `mdy`, `mdy`, `mdy`, `dmy`, `ymd`.Convierta las fechas anteriores a fecha, usando las respectivas funciones.

De texto

De texto

Por lo que basta con reconocer en qué orden están el año, mes y día, y acomodar las siglas correspondientes a la fecha que se está tratando.

De texto

Por lo que basta con reconocer en qué orden están el año, mes y día, y acomodar las siglas correspondientes a la fecha que se está tratando.

Cargue el archivo `Fechas.RData` que se le entregó, luego usando la función que corresponda, convierta la fecha de la tabla de Nacimientos a formato de fecha de lubridate. ¿Qué función usa?

De texto

Por lo que basta con reconocer en qué orden están el año, mes y día, y acomodar las siglas correspondientes a la fecha que se está tratando.

Cargue el archivo `Fechas.RData` que se le entregó, luego usando la función que corresponda, convierta la fecha de la tabla de Nacimientos a formato de fecha de lubridate. ¿Qué función usa? Repita lo anterior, pero usando la tabla de `Hijos`

De texto

Por lo que basta con reconocer en qué orden están el año, mes y día, y acomodar las siglas correspondientes a la fecha que se está tratando.

Cargue el archivo `Fechas.RData` que se le entregó, luego usando la función que corresponda, convierta la fecha de la tabla de Nacimientos a formato de fecha de lubridate. ¿Qué función usa? Repita lo anterior, pero usando la tabla de `Hijos`

```
load(file = 'Fechas.RData')
Fechas_Nacimiento <- Nacimientos %>%
  mutate(Fecha = ymd(Fecha))
Hijos <- Hijos %>%
  mutate(Nacimiento_Hijo = dmy(Nacimiento_Hijo))
str(Fechas_Nacimiento)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    3103 obs. o
##  $ ID       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Fecha    : Date, format: "1903-11-30" "1905-11-03" ...
```

Componentes individuales

Componentes individuales

Algunas veces las fechas están separadas, como es el caso de la tabla de fallecimientos

Para esto se usa la función `make_date`, que recibe como parámetros: `year`, `month` y `day`.

```
fecha8 <- make_date(year = 2017,month = 3,day = 22)
fecha8
```

```
## [1] "2017-03-22"
```

Componentes individuales

Algunas veces las fechas están separadas, como es el caso de la tabla de fallecimientos

Para esto se usa la función `make_date`, que recibe como parámetros: `year`, `month` y `day`.

```
fecha8 <- make_date(year = 2017,month = 3,day = 22)
fecha8
```

```
## [1] "2017-03-22"
```

¿Qué pasa si alguno(s) de los parámetros de `make_date` es de tipo character (string)?

Componentes individuales

Algunas veces las fechas están separadas, como es el caso de la tabla de fallecimientos

Para esto se usa la función `make_date`, que recibe como parámetros: `year`, `month` y `day`.

```
fecha8 <- make_date(year = 2017,month = 3,day = 22)
fecha8
```

```
## [1] "2017-03-22"
```

¿Qué pasa si alguno(s) de los parámetros de `make_date` es de tipo character (string)? En la tabla de fallecimientos, defina una columna que tenga la fecha como objeto date, usando la función `make_date`, y las columnas correspondientes.

Solución Fallecimientos

Solución Fallecimientos

```
Fallecimientos_f <- Fallecimientos %>%
  transmute(Fecha = make_date(year = Anno_Muerte,
                              month = Mes_Muerte,
                              day = Dia_Muerte),
            ID)
str(Fallecimientos_f)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    3103 obs. o
##  $ Fecha: Date, format: "1966-11-17" "1971-06-26" ...
##  $ ID   : int  1 2 3 4 5 6 7 8 9 10 ...
```

Componentes individuales

También se pueden manejar fechas junto con el tiempo en un mismo objeto, usando la función `make_datetime`:

```
tiempo1 <- make_datetime(year = 2017,month = 3,day = 22,
                         hour = 1,min = 53,sec = 10)
tiempo1
```

```
## [1] "2017-03-22 01:53:10 UTC"
```

Note que al final viene la zona horaria, que usa la zona `UTC` de forma predefinida.

Componentes individuales

También se pueden manejar fechas junto con el tiempo en un mismo objeto, usando la función `make_datetime`:

```
tiempo1 <- make_datetime(year = 2017,month = 3,day = 22,
                         hour = 1,min = 53,sec = 10)
tiempo1
```

```
## [1] "2017-03-22 01:53:10 UTC"
```

Note que al final viene la zona horaria, que usa la zona `UTC` de forma predefinida. Para saber en cuál zona horaria estamos, se usa `Sys.timezone()`. Si se quiere indicar, se puede usar en el parámetro `tz`.

Componentes individuales

También se pueden manejar fechas junto con el tiempo en un mismo objeto, usando la función `make_datetime`:

```
tiempo1 <- make_datetime(year = 2017,month = 3,day = 22,
                         hour = 1,min = 53,sec = 10)
tiempo1
```

```
## [1] "2017-03-22 01:53:10 UTC"
```

Note que al final viene la zona horaria, que usa la zona `UTC` de forma predefinida. Para saber en cuál zona horaria estamos, se usa `Sys.timezone()`. Si se quiere indicar, se puede usar en el parámetro `tz`. ¿En qué zona horaria está su computadora?

Componentes individuales

También se pueden manejar fechas junto con el tiempo en un mismo objeto, usando la función `make_datetime`:

```
tiempo1 <- make_datetime(year = 2017,month = 3,day = 22,
                         hour = 1,min = 53,sec = 10)
tiempo1
```

```
## [1] "2017-03-22 01:53:10 UTC"
```

Note que al final viene la zona horaria, que usa la zona `UTC` de forma predefinida. Para saber en cuál zona horaria estamos, se usa `Sys.timezone()`. Si se quiere indicar, se puede usar en el parámetro `tz`. ¿En qué zona horaria está su computadora?

Cree una columna con fecha **y** tiempo para la tabla de fallecimientos.

Solución fallecimientos

```
Fallecimientos <- Fallecimientos %>%
  transmute(Fecha_tiempo = make_datetime(
    year = Anno_Muerte,
    month = Mes_Muerte,
    day = Dia_Muerte,
    hour = Hora_Muerte,
    min = Minuto_Muerte),
              ID)
str(Fallecimientos)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    3103 obs. o
##  $ Fecha_tiempo: POSIXct, format: "1966-11-17 18:03:00"
##  $ ID          : int  1 2 3 4 5 6 7 8 9 10 ...
```

De otros tipos

Se puede usar la función `as_datetime` para convertir un objeto tipo fecha, a un objeto que incluya la hora.

```
as_datetime(fecha8)
```

```
## [1] "2017-03-22 UTC"
```

Y también se puede convertir un objeto tipo `POSIXct` (tiempo) a fecha, usando `as_date`

Otras formas de convertir:

Otras formas de convertir:

Figure 2

Otras formas de convertir:

Otras formas de convertir:

Además de las funciones tipo `ymd`, hay otras en las que se pueden especificar la hora, minuto y mes al objeto que se crea:

```
fecha_g <- ymd_hms('2016-10-20 15:35:20',
                   tz = 'America/Guatemala')
fecha_g_2 <- ymd_hm('2016-10-20 15:35',
                   tz = 'America/Guatemala')
fecha_g_3 <- ymd_h('2016-10-20 15',
                   tz = 'America/Guatemala')
```

Obtener los componentes

Obtener los componentes

Muchas veces no se ocupa toda la fecha, sino uno de sus componentes. Para esto se pueden usar las siguientes funciones:

función	Resultado
`year`	año
`month`	mes
`day`	día del mes
`mday`	día del mes
`yday`	día del año
`wday`	día de la semana

Obtener los componentes

Muchas veces no se ocupa toda la fecha, sino uno de sus componentes. Para esto se pueden usar las siguientes funciones:

función	Resultado
`year`	año
`month`	mes
`day`	día del mes
`mday`	día del mes
`yday`	día del año
`wday`	día de la semana

Las funciones `month.mday` tiene dos parámetros opcionales

Obtener los componentes

Muchas veces no se ocupa toda la fecha, sino uno de sus componentes. Para esto se pueden usar las siguientes funciones:

función	Resultado
`year`	año
`month`	mes
`day`	día del mes
`mday`	día del mes
`yday`	día del año
`wday`	día de la semana

Las funciones `month.mday` tiene dos parámetros opcionales
Compruebe el valor en `fecha_g` de las funciones anteriores. Haga un gráfico de barras (en ggplot), usando el día de la semana de las fechas de nacimiento.

Obtener los componentes

Muchas veces no se ocupa toda la fecha, sino uno de sus componentes. Para esto se pueden usar las siguientes funciones:

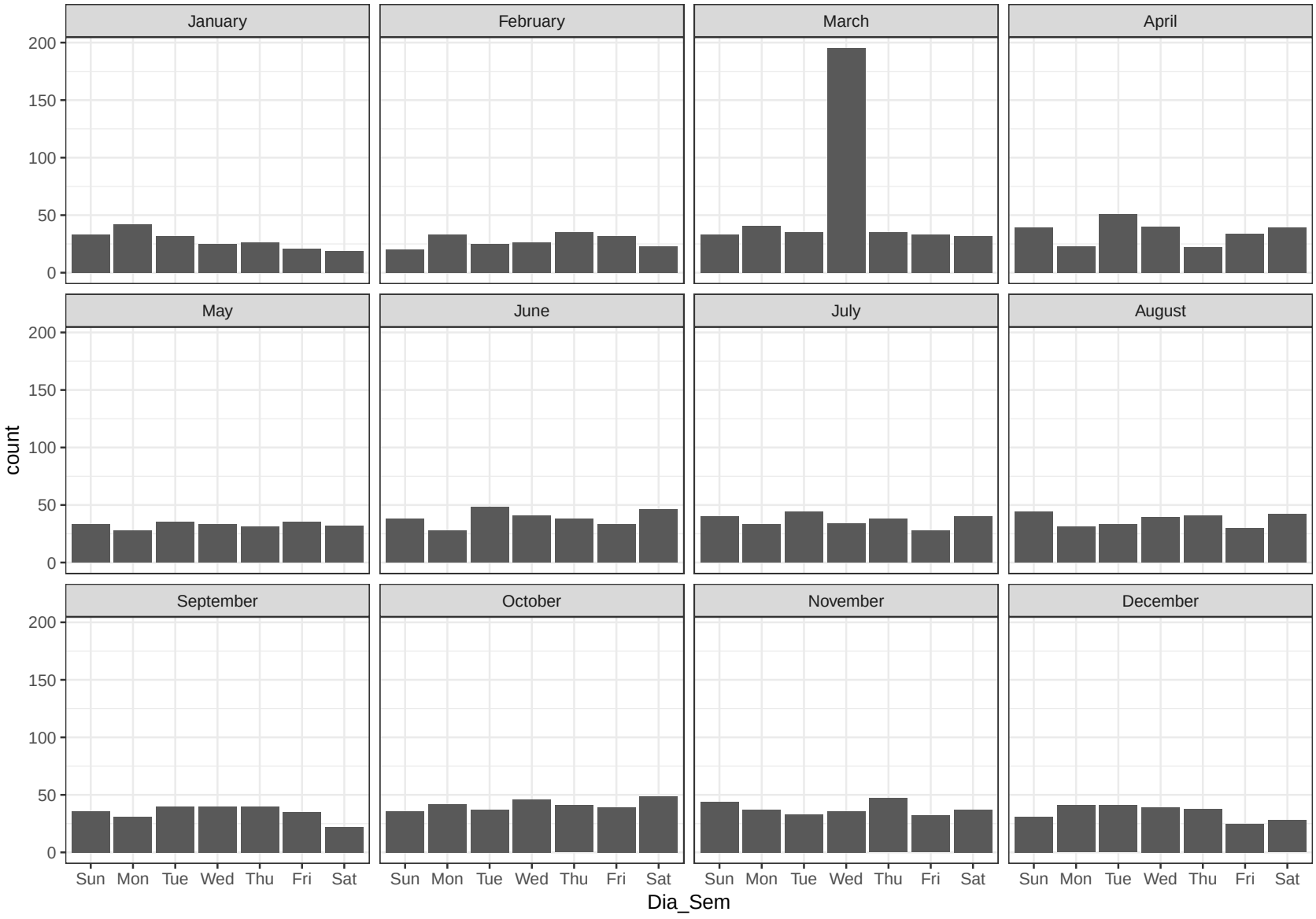
función	Resultado
`year`	año
`month`	mes
`day`	día del mes
`mday`	día del mes
`yday`	día del año
`wday`	día de la semana

Las funciones `month.mday` tiene dos parámetros opcionales
Compruebe el valor en `fecha_g` de las funciones anteriores. Haga un gráfico de barras (en ggplot), usando el día de la semana de las fechas de nacimiento. Repita el gráfico anterior pero usando un `facet_wrap` con los meses. ¿Concluye algo distinto o similar? ¿Qué otro gráfico podría hacer?

Solucion:

```
library(ggplot2)
Plot_Semana_Mes <- Fechas_Nacimiento %>%
  mutate(Dia_Sem = wday(Fecha,label = TRUE,abbr = TRUE),
         Mes = month(Fecha,label = TRUE,abbr = FALSE)) %>%
  ggplot(aes(Dia_Sem)) +
  geom_bar() +
  facet_wrap('Mes') +
  theme_bw()
```

Solucion


January
February
March
April
May
June
July
August
September
October
November
December
count
0
50
100
150
200
Sun
Mon
Tue
Wed
Thu
Fri
Sat
Dia_Sem

Redondear

Redondear

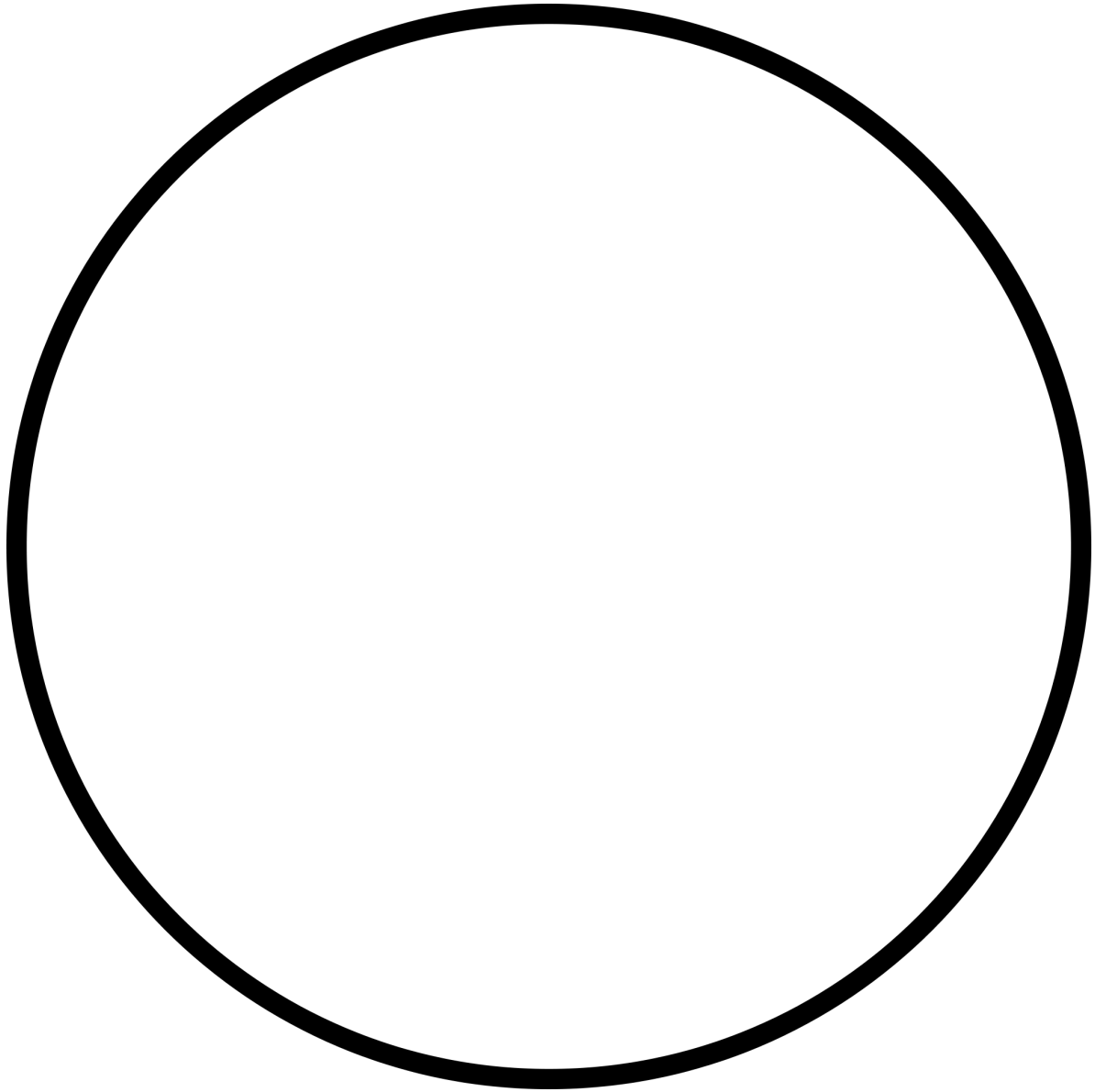


Figure 3

Redondear

Estas funciones reciben primero la fecha, y luego la unidad de tiempo (semana, mes, año, etc. . .) con la que se quiere efectuar el redondeo.

Las funciones son: `floor_date`, `round_date` y `ceiling_date`.

Entonces, si queremos saber cuál es el inicio del mes, hacemos:

```
floor_date(fecha_g,unit = 'month')
```

```
## [1] "2016-10-01 CST"
```

Redondear

Estas funciones reciben primero la fecha, y luego la unidad de tiempo (semana, mes, año, etc. . .) con la que se quiere efectuar el redondeo.

Las funciones son: `floor_date`, `round_date` y `ceiling_date`.

Entonces, si queremos saber cuál es el inicio del mes, hacemos:

```
floor_date(fecha_g,unit = 'month')
```

```
## [1] "2016-10-01 CST"
```

¿Qué comandos puede combinar para saber qué día comienza la semana?

Redondear

Estas funciones reciben primero la fecha, y luego la unidad de tiempo (semana, mes, año, etc. . .) con la que se quiere efectuar el redondeo.

Las funciones son: `floor_date`, `round_date` y `ceiling_date`.

Entonces, si queremos saber cuál es el inicio del mes, hacemos:

```
floor_date(fecha_g,unit = 'month')
```

```
## [1] "2016-10-01 CST"
```

¿Qué comandos puede combinar para saber qué día comienza la semana?

```
wday(floor_date(today(),unit = 'week'),label = TRUE)
```

Modificar componentes

También se pueden modificar los componentes de las fechas usando las mismas funciones que se mencionan antes:

```
year(fecha_g) <- 2014
fecha_g
```

```
## [1] "2014-10-20 15:35:20 CST"
```

Modificar componentes

También se pueden modificar los componentes de las fechas usando las mismas funciones que se mencionan antes:

```
year(fecha_g) <- 2014
fecha_g
```

```
## [1] "2014-10-20 15:35:20 CST"
```

```
month(fecha_g) <- 3
fecha_g
```

```
## [1] "2014-03-20 15:35:20 CST"
```

Modificar componentes

También se pueden modificar los componentes de las fechas usando las mismas funciones que se mencionan antes:

```
year(fecha_g) <- 2014
fecha_g
```

```
## [1] "2014-10-20 15:35:20 CST"
```

```
month(fecha_g) <- 3
fecha_g
```

```
## [1] "2014-03-20 15:35:20 CST"
```

```
yday(fecha_g) <- 10
fecha_g
```

```
## [1] "2014-01-10 15:35:20 CST"
```

update

update

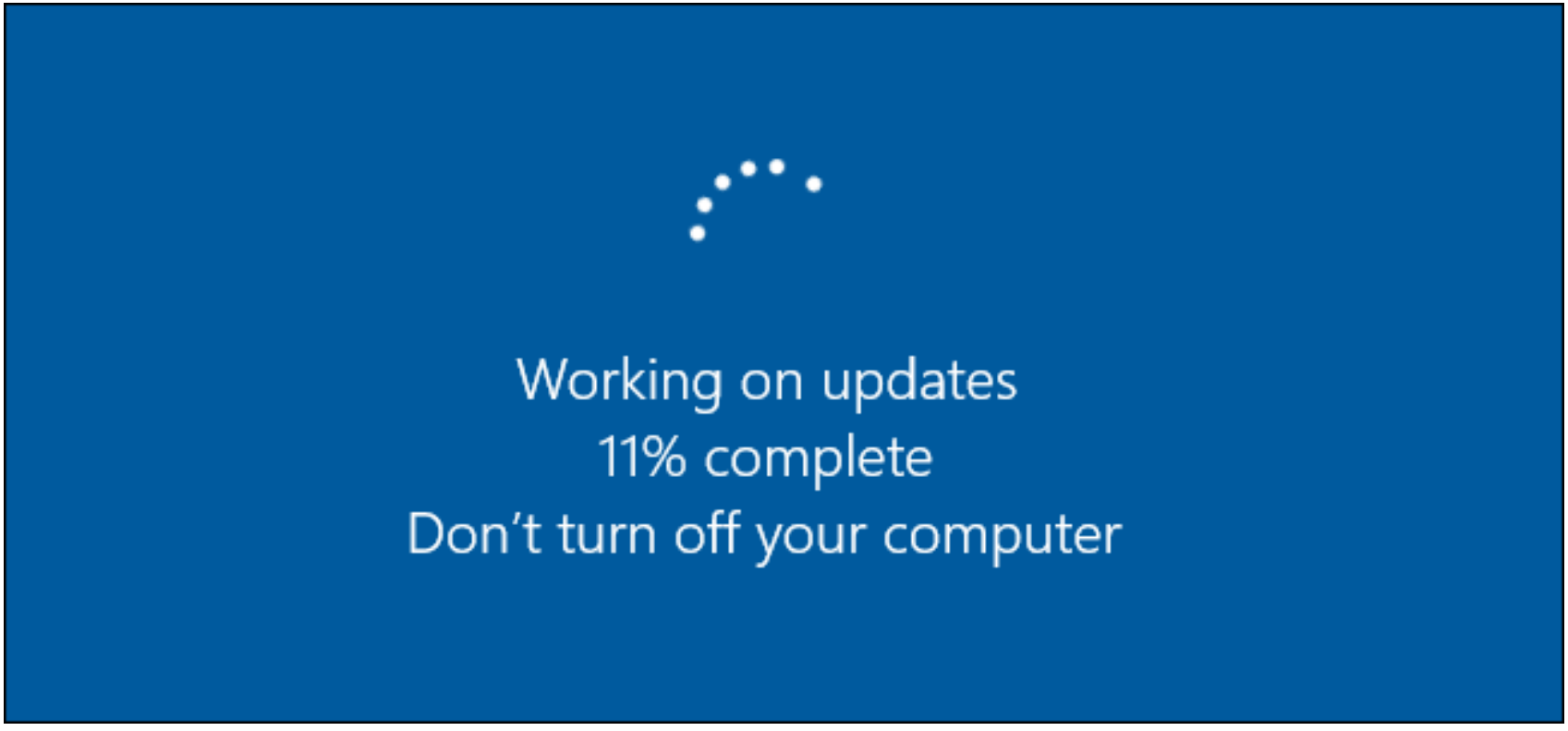


Figure 4: Por favor de Windows no

update

update

Esta función permite realizar cambios usando una fecha de referencia, sin cambiar la fecha "base". Por ejemplo:

```
fecha_2g <- update(fecha_g,month =2)
c(fecha_2g,fecha_g)
```

```
## [1] "2014-02-10 15:35:20 CST" "2014-01-10 15:35:20 CST"
```

update

Esta función permite realizar cambios usando una fecha de referencia, sin cambiar la fecha "base". Por ejemplo:

```
fecha_2g <- update(fecha_g,month =2)
c(fecha_2g,fecha_g)
```

```
## [1] "2014-02-10 15:35:20 CST" "2014-01-10 15:35:20 CST"
```

¿Cómo haría para saber cuál es el mes y año que tiene mayor cantidad de nacimientos de los hijos?

Solución

```
Hijos %>%
  mutate(Anno_mes = update(Nacimiento_Hijo,mday = 1)) %>%
  group_by(Anno_mes) %>%
  summarise(Total = n()) %>%
  arrange(Total) %>%
  tail(1)
```

```
## # A tibble: 1 x 2
##   Anno_mes   Total
##   <date>     <int>
## 1 1930-12-01  2865
```

Duraciones

Muchas veces uno quiere trabajar con intervalos de tiempo, una duración (`duration`) es lo que se obtiene al sacar la diferencia entre dos fechas:

```
tiempo_lectivo <- today() - ymd(20180813)
as.duration(tiempo_lectivo)
```

```
## [1] "7171200s (~11.86 weeks)"
```

La ventaja de las duraciones es que se pueden sumar (y restar) de forma cómoda con las fechas:

```
as.duration(tiempo_lectivo) + today()
```

```
## [1] "2019-01-26"
```

Duraciones

En el fondo, lo que hacen los objetos `duration` es que cuentan segundos, y a partir de esto operan con las fechas. Para esto, existen varios constructores de duraciones que son:

```
dseconds(c(46,99))
dminutes(31)
dhours(c(12,48))
ddays(c(2,62))
dweeks(15)
dyears(0:2)
```

Ejercicios:

- ¿Qué fecha fue hace: un año, 2 semanas y 96 horas?
- ¿Cuál es el ID que tuvo un hijo a la edad más joven?
- ¿Cuál es el ID que falleció teniendo el hijo más reciente?

Multiplicar duraciones

También se pueden multiplicar duraciones, lo cual va a multiplicar la cantidad de segundos que corresponden a esa duración.

```
today() + (dyears(1)+ ddays(25) - dweeks(7))*2
```

```
## Note: method with signature 'Duration#ANY' chosen for fu
##  target signature 'Duration#Duration'.
##  "ANY#Duration" would also be valid
```

```
## [1] "2020-09-16"
```

Fallas de duraciones

¿Qué pasa si nos devolvemos dos año desde el primero de enero?

Fallas de duraciones

¿Qué pasa si nos devolvemos dos año desde el primero de enero?

```
ymd(20180101) - dyears(2)
```

```
## [1] "2016-01-02"
```

¡Falla!

Fallas de duraciones

¿Qué pasa si nos devolvemos dos año desde el primero de enero?

```
ymd(20180101) - dyears(2)
```

```
## [1] "2016-01-02"
```

¡Falla! Pues el 2016 es año bisiesto, por lo que tiene 366 días.

Fallas de duraciones

¿Qué pasa si nos devolvemos dos año desde el primero de enero?

```
ymd(20180101) - dyears(2)
```

```
## [1] "2016-01-02"
```

¡Falla! Pues el 2016 es año bisiesto, por lo que tiene 366 días. Si tuviéramos horario de veranos también perderíamos una hora en ciertos días del año.

```
fecha_falla <- ymd_hms('20170312 00:00:00',
                       tz = 'America/New_York')
fecha_falla + ddays(1)
```

```
## [1] "2017-03-13 01:00:00 EDT"
```

Un día de 25 horas... Casual...

Periodos

Periodos

Periodos

Como vimos, las duraciones pueden fallar. Es por esto que se usan los periodos, que funcionan en tiempos de "humanos", y si mantienen la consistencia en este sentido. Se pueden usar con la función: `as.period`

```
as.period(tiempo_lectivo) + today()
```

```
## [1] "2019-01-26"
```

Se pueden definir, de forma similar a las duraciones, usando las funciones:

```
seconds(c(46,99))
minutes(31)
hours(c(12,48))
days(c(2,62))
weeks(15)
months(2)
years(0:2)
```

Comparemos:

```
fecha_falla + days(1)
```

```
## [1] "2017-03-13 EDT"
```

```
ymd(20170101) - years(1)
```

```
## [1] "2016-01-01"
```

Meses

Note que ahora **sí** tenemos meses. No se pueden usar como duraciones, pues no hay una cantidad "estándar" de duración para un mes.

Meses

Note que ahora **sí** tenemos meses. No se pueden usar como duraciones, pues no hay una cantidad "estándar" de duración para un mes.

Incluir los meses tiene un *costo* implícito, que es que no todas las fechas están bien definidas para un mes después. Por ejemplo, un mes después del 31 de marzo, puede ser:

Meses

Note que ahora **sí** tenemos meses. No se pueden usar como duraciones, pues no hay una cantidad "estándar" de duración para un mes.

Incluir los meses tiene un *costo* implícito, que es que no todas las fechas están bien definidas para un mes después. Por ejemplo, un mes después del 31 de marzo, puede ser:

1. 31 de abril, que no existe

Meses

Note que ahora **sí** tenemos meses. No se pueden usar como duraciones, pues no hay una cantidad "estándar" de duración para un mes.

Incluir los meses tiene un *costo* implícito, que es que no todas las fechas están bien definidas para un mes después. Por ejemplo, un mes después del 31 de marzo, puede ser:

1. 31 de abril, que no existe
2. 1ro de mayo, 31 días después

Meses

Note que ahora **sí** tenemos meses. No se pueden usar como duraciones, pues no hay una cantidad "estándar" de duración para un mes.

Incluir los meses tiene un *costo* implícito, que es que no todas las fechas están bien definidas para un mes después. Por ejemplo, un mes después del 31 de marzo, puede ser:

1. 31 de abril, que no existe
2. 1ro de mayo, 31 días después
3. 30 de abril

Meses

Solo la primera de estas opciones cumple el hecho de que $a + b - b = a$. Pero las otras opciones también están implementadas, o se puede obtener usando un poco de aritmética sencilla:

```
marzo31 <- ymd(20180331)
marzo31 + months(1)
```

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

```
marzo31 + days(31)
```

```
## [1] "2018-05-01"
```

```
marzo31 %m+% months(1) # Para restar se usa %m-%
```

```
## [1] "2018-04-30"
```

Hay que tener **mucho** cuidado con el orden, pues no lo preservan...

Intervalos

¿Cuántos días hay en un año?

Intervalos

¿Cuántos días hay en un año?

```
years(1)/days(1)
```

```
## estimate only: convert to intervals for accuracy
```

```
## [1] 365.25
```

Intervalos

¿Cuántos días hay en un año?

```
years(1)/days(1)
```

```
## estimate only: convert to intervals for accuracy
```

```
## [1] 365.25
```

Depende del año! Para esto se usan los intervalos, que guardan la información del inicio y del final del *intervalo* de tiempo.

Intervalos

Intervalos

Se pueden definir usando:

```
inv_1 <- interval(start = marzo31,
          end = today())
int_length(inv_1)
```

```
## [1] 18835200
```

Intervalos

Se pueden definir usando:

```
inv_1 <- interval(start = marzo31,
          end = today())
int_length(inv_1)
```

```
## [1] 18835200
```

Esto no es muy útil. Pero se puede convertir, ahora sí, a una duración o a un periodo:

```
as.period(inv_1)
```

```
## [1] "7m 4d 0H 0M 0S"
```

```
as.duration(inv_1)
```

```
## [1] "18835200s (~31.14 weeks)"
```

Intervalos

También, se puede definir usando:

```
inv_2 <- today() %--% marzo31
```

Intervalos

También, se puede definir usando:

```
inv_2 <- today() %--% marzo31
```

Hay funciones para accesar el inicio y el final de los intervalos, llamadas: `int_start` e `int_end`.

Intervalos

También, se puede definir usando:

```
inv_2 <- today() %--% marzo31
```

Hay funciones para accesar el inicio y el final de los intervalos, llamadas: `int_start` e `int_end`.

Note que acá se tiene que la fecha de hoy es el inicio del intervalo.

Intervalos

También, se puede definir usando:

```
inv_2 <- today() %--% marzo31
```

Hay funciones para accesar el inicio y el final de los intervalos, llamadas: `int_start` e `int_end`.

Note que acá se tiene que la fecha de hoy es el inicio del intervalo. Por lo que podemos hacer un `int_flip` con este intervalo para obtener el intervalo igual a `inv_1`

```
inv_2 <- int_flip(inv_2)
inv_2
```

```
## [1] 2018-03-31 UTC--2018-11-04 UTC
```

Intervalos

Si se quiere obtener en cierta unidad de tiempo, se puede hacer la división por una unidad de ese tiempo:

```
inv_2/days(1)
```

```
## [1] 218
```

```
inv_2/months(1)
```

```
## [1] 7.16129
```

```
inv_2/weeks(1)
```

```
## [1] 31.14286
```

Operaciones de intervalos

Para comparar intervalos se puede hacer: `int_overlaps`, para identificar si dos intervalos comparten al menos un segundo.

Para saber si comienzan o terminan en el mismo momento, se puede usar `int_align`.

Mover un intervalo

Para mover un intervalo se puede usar `int_shift`, indicando el intervalo y la cantidad de tiempo que se quiere correr:

```
int_shift(inv_2,by = days(1))
```

```
## [1] 2018-04-01 UTC--2018-11-05 UTC
```

```
int_shift(inv_2,by = months(1))
```

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

Mover un intervalo

Note que como `months` es una duración, al agregarlo a la fecha de inicio, queda una fecha inexistente. Por lo que deja de existir el intervalo. Sin embargo, si agregamos una duración sí se obtiene un valor:

```
int_shift(inv_2,by = duration(1,units = 'months'))
```

```
## [1] 2018-04-30 10:00:00 UTC--2018-12-04 10:00:00 UTC
```

Ejercicios

Haga las uniones entre los nacimientos de hijos, los padres y sus fallecimientos. Luego cree los objetos de intervalo que corresponden a:

- Intervalo que vivió antes de cada hijo
- Intervalo de vida
- Intervalo de vida de los hijos antes de la muerte del padre.
- Intervalo de tiempo del menor hijo al mayor hijo.

Ejercicio, dificil:

¿Cuantas personas tuvieron más de un hijo en un periodo de menos de 9 meses ($\sim$40 semanas)? ¿Qué implicaciones tiene esto para la vida de esa persona?

Sugerencia: Ordene las fechas de nacimiento por ID.