Diplomado de Análisis Estadístico usando R

Módulo 3:
Exploración de datos

Profesor: *Víctor Macías E.*

En estas notas trabajaremos con:

- Fechas y horas
- Factores
- Strings

Para obtener información adicional a la discutida en esta guía, se recomienda revisar:

https://www.rstudio.com/resources/cheatsheets/

1. Fechas y horas

- Para trabajar con fechas y horas usaremos el paquete ***lubridate***:

https://cran.r-project.org/web/packages/lubridate/lubridate.pdf

```
library(lubridate)
```

- La norma ISO 8601 especifica la notación estándar utilizada para representar fechas y horas:
  - Fechas representadas como *YYYY-MM-DD*
  - Horas representadas como *HH:MM:SS*
- Hay tres tipos de datos que capturan la dimensión temporal de los datos:
  - *date* es la fecha. Por ejemplo, 2020-09-28
  - *time* es una hora dentro de un día. Por ejemplo, 08:48:12
  - *date-time* combina *date* y *time*. Son también conocidos como *POSIXct*. Por ejemplo, 2020-09-28 08:48:12
- Una fecha se puede crear de:
  - Un *string*.
  - Componentes individuales de *date-time*.
  - Otra fecha o fecha-hora.

A continuación se presentan varios ejemplos de cómo crear un *date*, *time* o *date-time*:

```
as.Date("2020-09-27")   # forma 1
```

```
## [1] "2020-09-27"
```

```
ymd("2020-09-27")   # forma 2
```

```
## [1] "2020-09-27"
```

```r
ymd(20200927)   # forma 3 - Note que en esta forma no se usó un string
```

```
## [1] "2020-09-27"
```

```r
dmy("27-09-2020") # forma 4
```

```
## [1] "2020-09-27"
```

```r
as.POSIXct("2020-09-28 20:13:00", tz = "America/El_Salvador")  # forma 1
```

```
## [1] "2020-09-28 20:13:00 CST"
```

```r
ymd_hms("2020-09-21 14:04:32", tz = "America/El_Salvador")  # forma 2
```

```
## [1] "2020-09-21 14:04:32 CST"
```

```r
Sys.timezone()
```

```
## [1] "America/Santiago"
```

```r
dmy(31012017)   # forma 1
```

```
## [1] "2017-01-31"
```

```r
dmy("31-Enero-2017") # forma 2
```

```
## [1] "2017-01-31"
```

Para convertir un objeto a *date* o *datetime* usar *as_date()* o *as_datetime()*.

```r
as_date("2020-09-29 20:49:34")
```

```
## [1] "2020-09-29"
```

```r
as_datetime("2020-09-29")
```

```
## [1] "2020-09-29 UTC"
```

```r
as_date(0)  # En este caso 0 es un número ¿Qué representa?
```

```
## [1] "1970-01-01"
```

```r
as_date(7)
```

```
## [1] "1970-01-08"
```

```r
as_date(-2)
```

```
## [1] "1969-12-30"
```

- Objetos que son definidos como fechas son almacenados internamente como el número de días desde el 1 de enero de 1970. Al definir fechas se pueden realizar operaciones aritméticas entre éstas, como por ejemplo:

```r
today()
```

```
## [1] "2020-11-24"
```

```r
now()
```

```
## [1] "2020-11-24 11:06:45 -03"
```

```r
ymd("2020-11-18") + 2
```

```
## [1] "2020-11-20"
```

```r
ymd("2020-12-31") - ymd("2020-11-18")
```

```
## Time difference of 43 days
```

```r
ymd("2020-11-18") + weeks(5)
```

```
## [1] "2020-12-23"
```

```r
today() + 1:5*weeks(1)
```

```
## [1] "2020-12-01" "2020-12-08" "2020-12-15" "2020-12-22" "2020-12-29"
```

```r
dmy("31-12-2019") %m+% (1:12*months(1))
```

```
##  [1] "2020-01-31" "2020-02-29" "2020-03-31" "2020-04-30" "2020-05-31"
##  [6] "2020-06-30" "2020-07-31" "2020-08-31" "2020-09-30" "2020-10-31"
## [11] "2020-11-30" "2020-12-31"
```

- Hay una lista de 594 zonas horarias a nivel mundial, las cuales se pueden obtener OlsonNames(). En el caso del continente americano, la lista incluye 167 zonas horarias. Las primeras 10 son:

```
##  [1] "America/Adak"                  "America/Anchorage"
##  [3] "America/Anguilla"              "America/Antigua"
##  [5] "America/Araguaina"             "America/Argentina/Buenos_Aires"
##  [7] "America/Argentina/Catamarca"   "America/Argentina/ComodRivadavia"
##  [9] "America/Argentina/Cordoba"     "America/Argentina/Jujuy"
## [11] "America/Argentina/La_Rioja"    "America/Argentina/Mendoza"
## [13] "America/Argentina/Rio_Gallegos" "America/Argentina/Salta"
## [15] "America/Argentina/San_Juan"    "America/Argentina/San_Luis"
## [17] "America/Argentina/Tucuman"     "America/Argentina/Ushuaia"
## [19] "America/Aruba"                 "America/Asuncion"
## [21] "America/Atikokan"              "America/Atka"
## [23] "America/Bahia"                 "America/Bahia_Banderas"
## [25] "America/Barbados"              "America/Belem"
## [27] "America/Belize"                "America/Blanc-Sablon"
## [29] "America/Boa_Vista"             "America/Bogota"
```

Para chequear la zona horaria en que nos encontramos:

```r
Sys.timezone()
```

```
## [1] "America/Santiago"
```

2. Factors

- Se usan para representar variables categóricas. También permiten mostrar un vector de strings en un orden que no sea alfabético.
- Para trabajar con factores el paquete ***forcats*** es de gran ayuda:

https://cran.r-project.org/web/packages/forcats/forcats.pdf

```
library(forcats)
```

- Dos operaciones comunes realizadas con factores son:
  - Cambiar el orden de los niveles para lo cual puedes usar *fct_reorder()*
  - Cambiar los valores de los niveles usando *fct_recode()*

3. Strings

- Para trabajar con strings usaremos el paquete ***stringr***, cuya descripción se puede encontrar en la siguiente página:

https://cran.r-project.org/web/packages/stringr/stringr.pdf

```
library(stringr)
```

- Para crear strings usar comillas:

```
string1 <- "Veinte poemas de amor y una canción desesperada"
string2 <- 'Veinte poemas de amor y una canción desesperada'

class(string1)
```

```
## [1] "character"
```

```
class(string2)
```

```
## [1] "character"
```

- Entre las funciones que incluye el paquete `stringr` para manipular `strings` se incluyen:

Símbolo	Descripción
str_length	Retorna el número de caracteres de cada string
str_c	Combinar strings
str_sub	Extrae partes de un string
str_to_upper	Cambia a mayúsculas
str_to_lower	Cambia a minúsculas
str_detect	Entrega el valor=TRUE si el string contiene el patrón buscado y FALSE, si no lo encuentra
str_subset	Retorna los strings que contienen el patrón buscado
str_count	Retorna el número de veces que el patrón buscado ocurre en el string
str_replace	Reemplaza sólo la primera vez que ocurre el patrón
str_replace_all	Reemplaza cada vez que ocurre el patrón
str_trim	Elimina espacio en blanco al comienzo y/o final de un string

- Para manipular strings es muy útil aprender a trabajar con expresiones regulares *(regex)*, las que constituyen un modo de describir patrones específicos en strings. *En regex todos son caracteres.* Algunos ejemplos de símbolos usados en las expresiones regulares se presentan en la siguiente tabla:

Símbolo	Descripción
*	cero o más veces
+	uno o más veces
{n}	exactamente n veces
{n,}	al menos n veces
{,m}	como máximo m veces
{n,m}	entre n y m veces
^	Comienzo del string
$	Término del string
|	o
\\d o [0-9]	Cualquier dígito {0,1,2,3,4,5,6,7,8,9}
\\w	Cualquier caracter como letra, número o guión bajo
\\s	espacios en blanco (espacio, tab, nueva línea)
.	Cualquier caracter como número, letra, espacio en blanco
\\.	Símbolo de puntuación (punto seguido, punto aparte)
[a-zA-Z]	Todas las letras
[abc]	Matches letras a, b o c
[^abc]	Cualquier caracter excepto letras a,b,c
^\\d$	Comienzo string, seguido por un dígito, término del string