

# Tarea y Solución

## Pregunta 1

Investiga la documentación para decir cuales son los argumentos más importantes que trae la función `locale()`

### Solución

Directo de la documentación de R:

- **date\_names** Character representations of day and month names. Either the language code as string (passed on to `date_names_lang()`) or an object created by `date_names()`.
- **date\_format**, **time\_format** Default date and time formats.
- **decimal\_mark**, **grouping\_mark** Symbols used to indicate the decimal place, and to chunk larger numbers. Decimal mark can only be `,` or `..`
- **tz** Default tz. This is used both for input (if the time zone isn't present in individual strings), and for output (to control the default display). The default is to use "UTC", a time zone that does not use daylight savings time (DST) and hence is typically most useful for data. The absence of time zones makes it approximately 50x faster to generate UTC times than any other time zone. Use "" to use the system default time zone, but beware that this will not be reproducible across systems. For a complete list of possible time zones, see `OlsonNames()`. Americans, note that "EST" is a Canadian time zone that does not have DST. It is not Eastern Standard Time. It's better to use "US/Eastern", "US/Central" etc.
- **encoding** Default encoding. This only affects how the file is read - `readr` always converts the output to UTF-8.
- **asciiify** Should diacritics be stripped from date names and converted to ASCII? This is useful if you're dealing with ASCII data where the correct spellings have been lost. Requires the `stringi` package.

## Pregunta 2

- Investiga qué ocurre si intentamos configurar a la vez el **decimal\_mark** y **grouping\_mark** con el mismo carácter.
- ¿Qué valor por defecto toma el **grouping\_mark** cuando configuramos el **decimal\_mark** al carácter de coma “,”?
- ¿Qué valor por defecto toma el **decimal\_mark** cuando configuramos el **grouping\_mark** al carácter de punto “.”?

### Solución

- Error: **decimal\_mark** and **grouping\_mark** must be different.
- Quedan configurado al caracter contrario respectivamente.

### Pregunta 3

Investiga qué hace la opción del `locale()` cuando se utiliza junto al `date_format` y al `time_format` . Crea un ejemplo que muestre cuando puede sernos útil.

#### Solución

```
locale(date_names = "en",
       date_format = "%AD",
       time_format = "%AT",
       decimal_mark = ".",
       grouping_mark = ",",
       tz = "UTC",
       encoding = "UTF-8",
       asciify = FALSE)
```

### Pregunta 4

Crea un nuevo objeto locale que encapsule los ajustes más comunes de los parámetros para la carga de los fichero con los que sueles trabajar.

#### Solución

Usa la salida de la pregunta anterior para adaptar tus necesidades de valores para la función locale.

### Pregunta 5

Investiga las diferencias entre `read_csv()` y `read_csv2()`?

#### Solución

De la documentación oficial: `read_csv()` and `read_tsv()` are special cases of the general `read_delim()`. They're useful for reading the most common types of flat file data, comma separated values and tab separated values, respectively. `read_csv2()` uses ; for separators, instead of ,. This is common in European countries which use , as the decimal separator.

### Pregunta 6

Investiga las codificaciones que son más frecuentes en Europa y las más comunes en Asia. Usa un poco de Google e iniciativa para investigar acerca de este tema.

#### Solución

Prueba el siguiente enlace a la wikipedia: [https://en.wikipedia.org/wiki/Character\\_encoding](https://en.wikipedia.org/wiki/Character_encoding)

## Pregunta 7

Genera el formato correcto de string que procesa cada una de las siguientes fechas y horas:

```
v1 <- "May 19, 2018"  
v2 <- "2018-May-08"  
v3 <- "09-Jul-2013"  
v4 <- c("January 19 (2019)", "Mayo 1 (2015)")  
v5 <- "12/31/18" # Dic 31, 2014  
v6 <- "1305"  
v7 <- "12:05:11.15 PM"
```

## Solución

```
parse_date(v1, "%b %d, %Y")  
parse_date(v2, "%Y-%b-%d")  
parse_date(v3, "%d-%b-%Y")  
parse_date(v4, "%B %d (%Y)")  
parse_date(v5, "%m/%d/%y")  
parse_time(v6, format = "%H%M")  
parse_time(v7)
```