Tarea16

Yimmy Eman

2022-07-11

```
library(tidyverse)
```

Pregunta 1

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

- 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.
- asciify
  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.

```
#parse_number("1.000.000", locale = locale(grouping_mark = ".", decimal_mark = "."))
```

Error: `decimal_mark` and `grouping_mark` must be different

- ¿Qué valor por defecto toma el grouping_mark cuando configuramos el decimal_mark al carácter de coma “,”?

```
parse_number("1.000.000", locale = locale(grouping_mark = ".", decimal_mark = ","))
```

```
## [1] 1e+06
```

```
# Queda configurado al contrario
```

- ¿Qué valor por defecto toma el decimal_mark cuando configuramos el grouping_mark al carácter de punto “.”?

```
#parse_number("1.000.000", locale = locale(grouping_mark = ".", decimal_mark = "."))
```

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.

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

```
## <locale>
## Numbers:  123,456.78
## Formats:  %AD / %AT
## Timezone: UTC
## Encoding: UTF-8
## <date_names>
## Days:   Sunday (Sun), Monday (Mon), Tuesday (Tue), Wednesday (Wed), Thursday
##         (Thu), Friday (Fri), Saturday (Sat)
## Months: January (Jan), February (Feb), March (Mar), April (Apr), May (May),
##         June (Jun), July (Jul), August (Aug), September (Sep), October
##         (Oct), November (Nov), December (Dec)
## AM/PM:  AM/PM
```

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.

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

```
## <locale>
## Numbers:  123.456,78
## Formats:  %AD / %AT
## Timezone: UTC
## Encoding: UTF-8
## <date_names>
## Days:   domingo (dom.), lunes (lun.), martes (mar.), miércoles (mié.), jueves
##         (jue.), viernes (vie.), sábado (sáb.)
## Months: enero (ene.), febrero (feb.), marzo (mar.), abril (abr.), mayo (may.),
##         junio (jun.), julio (jul.), agosto (ago.), septiembre (sept.),
##         octubre (oct.), noviembre (nov.), diciembre (dic.)
## AM/PM:  a. m./p. m.
```

Pregunta 5

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

read_csv() y read_tsv() son casos especiales del read_delim() más general. Son útiles para leer los tipos más comunes de datos de archivos sin formato, valores separados por comas y valores separados por tabulaciones, respectivamente. read_csv2() utiliza ?;? para el separador de campos y ?,? para el punto decimal. Este formato es común en algunos países europeos.

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.

Pregunta 7

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

```
v1 <- "May 19, 2018"
parse_date(v1, "%b %d, %Y")
```

```
## [1] "2018-05-19"
```

```
v2 <- "2018-May-08"
parse_date(v2, "%Y-%b-%d")
```

```
## [1] "2018-05-08"
```

```
v3 <- "09-Jul-2013"
parse_date(v3, "%d-%b-%Y")
```

```
## [1] "2013-07-09"
```

```
v4 <- c("January 19 (2019)", "May 1 (2015)")
parse_date(v4, "%B %d (%Y)")
```

```
## [1] "2019-01-19" "2015-05-01"
```

```
v5 <- "12/31/18" # Dic 31, 2014
parse_date(v5, "%m/%d/%y")
```

```
## [1] "2018-12-31"
```

```
v6 <- "1305"
parse_time(v6, format = "%H%M")
```

```
## 13:05:00
```

```
v7 <- "12:05:11.15 PM"
parse_time(v7)
```

```
## 12:05:11
```