Numbers into English Words

Bill Venables

2020-01-26

Genesis and development

In answer to a question on R-help John Fox provided an elegant R function to translate integers into English numbers. The present package extends this code to an S3 class, with constructor functions and methods to make this original idea more conveniently available.

The function `as.english` is intended to provide a parallel facility to the function `as.roman` in the `utils` package.

The main purpose of the package is to present an interesting programming example rather than to solve a likely real problem, though there could well be some applications in unusual contexts.

Note added in Version 1.1-4. The two small helper functions `words` and `Words` are included to facilitate inline code inserts in R markdown files. See the help files for examples. The `ordinal` function produces character strings and may be used directly in inline code inserts. Use `` `r words(103)` `` rather than `` `r english(103)` `` in R markdown files.

Note added in Version 1.2-0. The function `indefinite` added for algorithmically including an apprporiate indefinite article in a document insert. Based on a suggestion of Anne Pier Salverda.

The capitalised version, `Indefinite`, was added in 1.2-1.

This vignette was added in 1.2-2.

Examples

The main use envisaged for these tools is for code inserts in either R markdown or R noweb documents. It is easier here to demonstrate their function in plain code chunks, as below.

The resolution of an integer into words depends on the style of English chosen. English (UK) style inserts the conjunction *and* in various places where American (USA) style omits it. Wheter or not this happens is governed by the `english.UK` option, (a logical variable, `TRUE` for UK English), but if this option is unset, a guess is made as to which style is needed depending on the user's locale.

```
soldiers <- 10006
oldOpt <- options(english.UK = TRUE)
cat("The Duke of York had approximately", words(soldiers), "men.\n")
cat("How many did you say?  ", Words(soldiers), ", approximately.\n", sep = "")
The Duke of York had approximately ten thousand and six men.
How many did you say?  Ten thousand and six, approximately.
```

```
options(english.UK = FALSE)
cat("The Duke of York had approximately", words(soldiers), "men.\n")
cat("How many did you say?  ", Words(soldiers), ", approximately.\n", sep = "")
options(oldOpt)
The Duke of York had approximately ten thousand six men.
How many did you say?  Ten thousand six, approximately.
```

Ordinal numbers are handled as well.

```
days <- 1:6
cat(paste("\nOn the", ordinal(days), "day of Christmas..."))

On the first day of Christmas...
On the second day of Christmas...
On the third day of Christmas...
On the fourth day of Christmas...
On the fifth day of Christmas...
On the sixth day of Christmas...
```

The indefinite article, "a" or "an", can be algorithmically chosen and prepended, either in lower case or capitalised:

```
steps <- 7:11
cat(paste0("\nThis is ", indefinite(steps), "-step process..."))

This is a seven-step process...
This is an eight-step process...
This is a nine-step process...
This is a ten-step process...
This is an eleven-step process...
cat(paste0("\nThis is ", indefinite(steps, words = FALSE), "-step process..."))

This is a 7-step process...
This is an 8-step process...
This is a 9-step process...
This is a 10-step process...
This is an 11-step process...
cat(paste0("\n", Indefinite(ordinal(steps)), " step of the process is..."))

A seventh step of the process is...
An eighth step of the process is...
A ninth step of the process is...
A tenth step of the process is...
An eleventh step of the process is...
```

Roman notation is provided by the `utils::as.roman` function in a similar way:

```
numbers <- c(1:10, 1999, 2019)
punct <- c(rep(",", 10), " and",".")
cat(c("In Roman notation:",
      paste0("\n\t", Words(numbers), " is written as \"",
             utils::as.roman(numbers),"\"", punct)))
cat("\nDoing arithmetic in Roman notation can be difficult.")
In Roman notation:
    One is written as "I",
    Two is written as "II",
    Three is written as "III",
    Four is written as "IV",
    Five is written as "V",
    Six is written as "VI",
    Seven is written as "VII",
    Eight is written as "VIII",
    Nine is written as "IX",
    Ten is written as "X",
    One thousand nine hundred and ninety-nine is written as "MCMXCIX" and
```

```
    Two thousand and nineteen is written as "MMXIX".
Doing arithmetic in Roman notation can be difficult.
```

Some arithmetic operations are allowed for `english` objects, but to make sense the result should be in whole numbers.

```
english(100) + (-5):5
 [1] ninety-five           ninety-six            ninety-seven
 [4] ninety-eight          ninety-nine           one hundred
 [7] one hundred and one   one hundred and two   one hundred and three
[10] one hundred and four  one hundred and five
```