Package ‘humaniformat’

August 29, 2016

Title A Parser for Human Names
Version 0.6.0
Date 2016-04-24
Author Oliver Keyes [aut, cre]
Maintainer Oliver Keyes <ironholds@gmail.com>
Description Human names are complicated and nonstandard things. Humaniformat, which is based on Anthony Ettinger's 'humanparser' project (https://github.com/chovy/humanparser) provides functions for parsing human names, making a best-guess attempt to distinguish sub-components such as prefixes, suffixes, middle names and salutations.
License MIT + file LICENSE
LazyData true
URL https://github.com/ironholds/humaniformat/
BugReports https://github.com/ironholds/humaniformat/issues
Suggests testthat, knitr
LinkingTo Rcpp
Imports Rcpp, methods
VignetteBuilder knitr
RoxygenNote 5.0.1
NeedsCompilation yes
Repository CRAN
Date/Publication 2016-04-24 23:15:03

R topics documented:

  first_name ................................................. 2
  format_period ........................................... 3
  format_reverse .......................................... 3
  humaniformat ............................................ 4
first_name

last_name ............................................................ 4
middle_name .......................................................... 5
parse_names ............................................................. 6
salutation ................................................................. 6
suffix ................................................................. 7

Index 9

first_name  Get or set a name’s first name

Description
as in the lubridate package, individual components of a name can be both extracted or set using the relevant function call - see the examples.

Usage
first_name(x)

first_name(x) <- value

Arguments
x a name, or vector of names
value a replacement value for x’s first name.

See Also
salutation, middle_name, last_name and suffix for other accessors.

Examples
#Get a first name
eexample_name <- "Mr Jim Jeffries"
first_name(example_name)

#Set a first name
first_name(example_name) <- "Prof"
format_period

Reformat Period-Separated Names

Description

A common pattern for names is for first and middle names to be represented by initials. Unfortunately depending on how this is done, that can make things problematic; "G. K. Chesterton" is easy to parse, but "G.K. Chesterton" or "G.K.Chesterton" is not. format_period takes names that are period-separated in this fashion and reformats them to ensure there are spaces between each initial. Periods after any space in the name are preserved, so "G.K. Chesterton, M.D." does not become "G. K. Chesterton, M. D. ".

Usage

format_period(names)

Arguments

names a vector of names following this convention. Names that lack periods will be returned entirely intact, so assuming you don’t have (legitimate) periods in names not following this format, there’s no need to worry if your vector has mixed formatting.

See Also

format_reverse for names stored as "Lastname, Firstname", and parse_names to parse the output of this function.

Examples

format_period("G.K.Chesterton")

format_reverse

Reformat Reversed Names

Description

A common pattern for names is ‘Lastname Suffix, Salutation Firstname’ - or to put that more practically, ‘Jeffries PhD, Mr Bernard’. format_reverse takes these reversed names and reformats them to a form that parse_names can handle.

Usage

format_reverse(names)
Arguments

names a vector of names following this convention. Names that lack commas will be returned entirely intact, so assuming you don’t have (legitimate) commas in names not following this format, there’s no need to worry if your vector has mixed formatting.

Value

a vector containing the reformatted names

See Also

parse_names, which works more reliably if reversed names have been reformatted, and format_period for period-separated names.

Examples

# Take a reversed name and un-reverse it
format_reverse("Keyes, Oliver")

humaniformat  A Parser for Human Names

Description

Human names are complicated and nonstandard things. Humaniformat attempts to provide functions for parsing those names, making a best-guess attempt to distinguish sub-components such as prefixes, suffixes, middle names and salutations.

last_name  Get or set a name’s last name

Description

as in the lubridate package, individual components of a name can be both extracted or set using the relevant function call - see the examples.

Usage

last_name(x)

last_name(x) <- value
middle_name

Arguments

x a name, or vector of names
value a replacement value for x’s last name.

See Also

salutation, first_name, middle_name and suffix for other accessors.

Examples

# Get a last name
example_name <- "Mr Jim Toby Jeffries"
last_name(example_name)

# Set a last name
last_name(example_name) <- "Smith"

table

middle_name Get or set a name’s middle name

Description

As in the lubridate package, individual components of a name can be both extracted or set using the relevant function call - see the examples.

Usage

middle_name(x)

middle_name(x) <- value

Arguments

x a name, or vector of names
value a replacement value for x’s middle name.

See Also

salutation, first_name, last_name and suffix for other accessors.

Examples

# Get a middle name
example_name <- "Mr Jim Toby Jeffries"
middle_name(example_name)

# Set a middle name
middle_name(example_name) <- "Richard"
**parse_names**

*Parse Human Names*

**Description**

Human names are complex things; sometimes people have honorifics, or not. Or a single middle name, or many. Or a compound surname, or not a compound surname but 'PhD' at the end of their name, and augh.

parse_names provides a simple function for taking consistently formatted human names and splitting them into salutation, first_name, middle_name, last_name and suffix. It is capable of dealing with compound surnames, multiple middle names, and similar variations, and is fully vectorised.

**Usage**

```r
parse_names(names)
```

**Arguments**

- `names`: a character vector of names to parse.

**Value**

A data.frame with the columns salutation, first_name, middle_name, last_name, suffix and full_name (which contains the original name). In the event that a name doesn’t have a salutation, middle name, suffix, or so on, an NA will appear.

**Examples**

```r
# Parse a simple name
parse_names("Oliver Keyes")

# Parse a more complex name
parse_names("Hon. Oliver Timothy Keyes Esq.")
```

**salutation**

*Get or set a name's salutation*

**Description**

As in the lubridate package, individual components of a name can be both extracted or set using the relevant function call - see the examples. In the event that you attempt to set a component to NA, no modification will be made; in the event that you try to get a component that isn’t present, an NA will be returned.
suffix

Usage

  salutation(x)

  salutation(x) <- value

Arguments

  x  a name, or vector of names
  value  a replacement value for x’s salutation

See Also

  first_name, middle_name, last_name and suffix for other accessors.

Examples

  #Get a salutation
  example_name <- "Mr Jim Jeffries"
  salutation(example_name)

  #Set a salutation
  salutation(example_name) <- "Prof"


suffix  Get or set a name’s suffix

Description

  as in the lubridate package, individual components of a name can be both extracted or set using the relevant function call - see the examples.

Usage

  suffix(x)

  suffix(x) <- value

Arguments

  x  a name, or vector of names
  value  a replacement value for x’s suffix.

See Also

  salutation, first_name, middle_name and last_name for other accessors.
Examples

#Get a suffix
example_name <- "Mr Jim Toby Jeffries Esq"
suffix(example_name)

#Set a suffix
suffix(example_name) <- "PhD"
Index

first_name, 2, 5, 7
first_name<-(first_name), 2
format_period, 3, 4
format_reverse, 3, 3

humaniformat, 4
humaniformat-package (humaniformat), 4

last_name, 2, 4, 5, 7
last_name<-(last_name), 4

middle_name, 2, 5, 5, 7
middle_name<-(middle_name), 5

parse_names, 3, 4, 6

salutation, 2, 5, 6, 7
salutation<-(salutation), 6
suffix, 2, 5, 7, 7
suffix<-(suffix), 7