Package 'assertive.data'

November 21, 2018

Type Package

Title Assertions to Check Properties of Data

Version 0.0-3

Date 2018-11-20

Author Richard Cotton [aut, cre]

Maintainer Richard Cotton <richierocks@gmail.com>

Description A set of predicates and assertions for checking the properties of (country independent) complex data types. This is mainly for use by other package developers who want to include run-time testing features in their own packages. End-users will usually want to use assertive directly.

URL https://bitbucket.org/richierocks/assertive.data

BugReports https://bitbucket.org/richierocks/assertive.data/issues

Depends R (>= 3.0.0)

Imports assertive.base (>= 0.0-2), assertive.strings

Suggests testthat

License GPL (>= 3)

LazyLoad yes

LazyData yes

Acknowledgments Development of this package was partially funded by the Proteomics Core at Weill Cornell Medical College in Qatar <http://qatar-weill.cornell.edu>. The Core is supported by 'Biomedical Research Program' funds, a program funded by Qatar Foundation.

Collate 'imports.R' 'assert-is-data.R' 'is-data.R'

RoxygenNote 6.1.0

NeedsCompilation no

Repository CRAN

Date/Publication 2018-11-21 22:20:03 UTC

R topics documented:

`assert_all_are_cas_numbers`

Does the character vector contain CAS registry numbers?

Description

Checks that the input contains Chemical Abstract Service registry numbers.

Usage

```
assert_all_are_cas_numbers(x, na_ignore = FALSE,
  severity = getOption("assertive.severity", "stop"))

assert_any_are_cas_numbers(x, na_ignore = FALSE,
  severity = getOption("assertive.severity", "stop"))

is_cas_number(x)
```

Arguments

`x`	Input to check.
`na_ignore`	A logical value. If `FALSE`, `NA` values cause an error; otherwise they do not. Like `na.rm` in many stats package functions, except that the position of the failing values does not change.
`severity`	How severe should the consequences of the assertion be? Either `"stop"`, `"warning"`, `"message"`, or `"none"`.

Value

A logical vector that is `TRUE` when the input contains valid CAS registry numbers.

Note

CAS numbers take the form of 1 to 7 digits followed by a hyphen, followed by 2 digits, another hyphen and a final check digit.

References

Chemspider (https://www.chemspider.com) is a good service for looking up CAS numbers.

Examples

```
x <- c(
  water            = "7732-18-5",
  d_glucose        = "50-99-7",
  l_glucose        = "921-60-8",
  no_hyphens       = "7732185",
  two_check_digits = "7732-18-55",
  bad_check_digit  = "7732-18-4",
  missing          = NA
)
is_cas_number(x)
assert_any_are_cas_numbers(x)
#These examples should fail.
assertive.base::dont_stop(assert_all_are_cas_numbers(x))
```

---

assert_all_are_credit_card_numbers

Does the character vector contain credit card numbers?

Description

Checks that the input contains credit card numbers.

Usage

```
assert_all_are_credit_card_numbers(x, na_ignore = FALSE,
  severity = getOption("assertive.severity", "stop"))

assert_any_are_credit_card_numbers(x, na_ignore = FALSE,
  severity = getOption("assertive.severity", "stop"))

is_credit_card_number(x, type = c("visa", "mastercard", "amex", "diners",
  "discover", "jcb"))
```

Arguments

x	Input to check.
na_ignore	A logical value. If FALSE, NA values cause an error; otherwise they do not. Like na.rm in many stats package functions, except that the position of the failing values does not change.
severity	How severe should the consequences of the assertion be? Either "stop", "warning", "message", or "none".
type	Type of credit card. Multiple types can be selected.

Value

A logical vector that is TRUE when the input contains valid credit card numbers.

Note

Legacy card numbers, for example 13 digit Visa numbers and 15 digits JCB numbers are not supported.

References

http://www.regular-expressions.info/creditcard.html contains the regexes used by this function. The example valid card numbers were from PayPal developer documentation. The URL is no longer available.

Examples

```
x <- c(
  #visa
  "4111 1111 1111 1111",    #spaces are allowed where they
                            #would occur on the card
  "4012888888881881",       #though they can be omitted
  "4111 1111 1111 11111",   #too many digits
  "4012888888881882",       #bad check digit
  #mastercard
  "5555 5555 5555 4444",
  "5105 1051 0510 5100",
  "5655 5555 5555 4443",    #starts 56
  "51051 051 0510 5100",    #bad spacing
  #amex
  "3782 822463 10005",
  "3714 496353 98431",
  "3787 344936 71000",
  "3782 822463 1005",       #not enough digits
  #diners
  "3056 930902 5904",
  "3852 000002 3237",
  #discover
  "6011 1111 1111 1117",
  "6011 0009 9013 9424",
  #jcb
  "3530 1113 3330 0000",
  "3566 0020 2036 0505"
)
is_credit_card_number(x)
assert_any_are_credit_card_numbers(x)
assertive.base::dont_stop(assert_all_are_credit_card_numbers(x))
```

`assert_all_are_email_addresses` *Does the character vector contain email addresses?*

Description

Checks that the input contains email addresses. (It does not check the the address exists, merely that the string is in a suitable format.)

Usage

```
assert_all_are_email_addresses(x, method = c("simple", "rfc5322"),
  na_ignore = FALSE, severity = getOption("assertive.severity",
  "stop"))

assert_any_are_email_addresses(x, method = c("simple", "rfc5322"),
  na_ignore = FALSE, severity = getOption("assertive.severity",
  "stop"))

is_email_address(x, method = c("simple", "rfc5322"))
```

Arguments

`x`	Input to check.
`method`	Name of method to check for validity. See notes below.
`na_ignore`	A logical value. If `FALSE`, `NA` values cause an error; otherwise they do not. Like `na.rm` in many stats package functions, except that the position of the failing values does not change.
`severity`	How severe should the consequences of the assertion be? Either `"stop"`, `"warning"`, `"message"`, or `"none"`.

Value

A logical vector that is `TRUE` when the input contains valid email addresses.

Note

Each method specifies a regular expression (see `regex`) to match against. The `simple` method matches most email addresses in use, and is quite good at filtering out typos and nonsense. It won't match *every* email address however. For example, emails from a top level domain longer than 4 characters won't pass. The `rfc5322` method implements the official standard for emails. Thus all genuine emails will pass, but since the spec is very broad, it isn't as good at filtering out nonsense.

References

http://www.regular-expressions.info/email.html contains the regexes used by this function and a good discussion of the pros and cons of each.

Examples

```
addresses <- c(
  ok       = "a@b.com",
  no_at    = "a_at_b.com",
  no_dot   = "a@bcom",
  long_tld = "a@b.comma",
  punct    = "a!$&@b.com",
  missing  = NA
)
is_email_address(addresses)
is_email_address(addresses, method = "rfc5322")
```

`assert_all_are_hex_colors` *Does the character vector contain hex colors?*

Description

Checks that the input contains hexadecimal colors.

Usage

```
assert_all_are_hex_colors(x, na_ignore = FALSE,
  severity = getOption("assertive.severity", "stop"))

assert_any_are_hex_colors(x, na_ignore = FALSE,
  severity = getOption("assertive.severity", "stop"))

assert_all_are_hex_colours(x, na_ignore = FALSE,
  severity = getOption("assertive.severity", "stop"))

assert_any_are_hex_colours(x, na_ignore = FALSE,
  severity = getOption("assertive.severity", "stop"))

is_hex_color(x)

is_hex_colour(x)
```

Arguments

`x`	Input to check.
`na_ignore`	A logical value. If `FALSE`, `NA` values cause an error; otherwise they do not. Like `na.rm` in many stats package functions, except that the position of the failing values does not change.
`severity`	How severe should the consequences of the assertion be? Either `"stop"`, `"warning"`, `"message"`, or `"none"`.

Value

A logical vector that is TRUE when the input contains hex colours.

Note

A string is considered to represent a hexadecimal colour when contains a hash followed by six hex values. That is, digits or the letters from a to f (case insensitive).

Examples

```
x <- c(
  "#012345", "#789abc", "#defDEF",  #ok
  "012345",                         #no hash
  "#g12345",                        #bad letter
  "#01 23 45",                      #contains spaces
  "#12345", "#1234567"              #wrong length
)
is_hex_color(x)
assert_any_are_hex_colors(x)
#These examples should fail.
assertive.base::dont_stop(assert_all_are_hex_colors(x))
```

---

assert_all_are_honorifics

Is the string an honorific?

Description

Checks that the input contains honorifics (a.k.a. titles or salutations).

Usage

```
assert_all_are_honorifics(x, na_ignore = FALSE,
  severity = getOption("assertive.severity", "stop"))

assert_any_are_honorifics(x, na_ignore = FALSE,
  severity = getOption("assertive.severity", "stop"))

is_honorific(x)
```

Arguments

x	Input to check.
na_ignore	A logical value. If FALSE, NA values cause an error; otherwise they do not. Like na.rm in many stats package functions, except that the position of the failing values does not change.
severity	How severe should the consequences of the assertion be? Either "stop", "warning", "message", or "none".

Value

`is_honorific` returns `TRUE` if the input string contains a valid UK postcode. The assert_* function returns nothing but throws an error when the `is_*` function returns `FALSE`.

Note

Single full stops (periods) following a word boundary and preceding a space or the end of the string are stripped. Case is ignored. There is no formal list of official salutations, so this should only be used as a guide, rather than giving a definitive result. Especially note that cultural conventions differ across the world and this function has a UK bias.

References

Many possibilities borrowed from the Salutation dropdown on the MathWorks account creation page. https://www.mathworks.com/accesslogin/createProfile.do

Examples

```
x <- c("Mr", "MR", "mr.", "Mister", "masTer", "Mr!", "M.r", ".Mr")
is_honorific(x)
```

---

assert_all_are_ip_addresses

Does the character vector contain IP addresses?

Description

Checks that the input contains IP addresses. (It does not check that the address exists, merely that the string is in a suitable format.)

Usage

```
assert_all_are_ip_addresses(x, na_ignore = FALSE,
  severity = getOption("assertive.severity", "stop"))

assert_any_are_ip_addresses(x, na_ignore = FALSE,
  severity = getOption("assertive.severity", "stop"))

is_ip_address(x)
```

Arguments

`x`	Input to check.
`na_ignore`	A logical value. If `FALSE`, `NA` values cause an error; otherwise they do not. Like `na.rm` in many stats package functions, except that the position of the failing values does not change.
`severity`	How severe should the consequences of the assertion be? Either `"stop"`, `"warning"`, `"message"`, or `"none"`.

Value

A logical vector that is TRUE when the input contains valid IP addresses.

Note

Valid IP addresses are considered to be four integers in the range 0 to 255, separated by dots, or the string "localhost".

Examples

```
x <- c(
  localhost     = "localhost",
  valid_address = "255.0.255.0",
  out_of_range  = "1.2.3.256",
  five_blocks   = "1.2.3.4.5",
  non_numeric   = "1.2.3.Z",
  missing_block = "1.2.3.NA",
  missing       = NA
)
is_ip_address(x)
assert_any_are_ip_addresses(x)
#These examples should fail.
assertive.base::dont_stop(assert_all_are_ip_addresses(x))
```

---

assert_all_are_isbn_codes

Does the character vector contain ISBN book codes?

Description

Checks that the input contains ISBN-10 or ISBN-13 book codes.

Usage

```
assert_all_are_isbn_codes(x, type = c("10", "13"), na_ignore = FALSE,
  severity = getOption("assertive.severity", "stop"))

assert_any_are_isbn_codes(x, type = c("10", "13"), na_ignore = FALSE,
  severity = getOption("assertive.severity", "stop"))

is_isbn10_code(x, .xname = get_name_in_parent(x))

is_isbn13_code(x, .xname = get_name_in_parent(x))

is_isbn_code(x, type = c("10", "13"))
```

Arguments

x	Input to check.
type	Either "isbn10", "isbn13" or both (for matching either type).
na_ignore	A logical value. If FALSE, NA values cause an error; otherwise they do not. Like na.rm in many stats package functions, except that the position of the failing values does not change.
severity	How severe should the consequences of the assertion be? Either "stop", "warning", "message", or "none".
.xname	Not intended to be called directly.

Value

A logical vector that is TRUE when the input contains valid ISBN book codes.

Examples

```
x10 <- c(
  hyphens             = "0-387-98503-4",
  spaces              = "0 387 98503 4",
  just_numbers        = "0387985034",
  too_long            = "00-387-98503-4",
  too_short           = "0-387-9850-4",
  non_numeric         = "Z-387-98503-4",
  invalid_check_digit = "0-387-98503-5",
  missing             = NA
)
x13 <- c(
  hyphens             = "978-0-387-98503-9",
  spaces              = "978 0 387 98503 9",
  just_numbers        = "9780387985039",
  too_long            = "9978-0-387-98503-9",
  too_short           = "978-0-387-9850-9",
  non_numeric         = "Z78-0-387-9850-9",
  invalid_check_digit = "978-0-387-98503-8",
  missing             = NA
)
is_isbn_code(x10, type = "10")
assert_any_are_isbn_codes(x10, type = "10")
is_isbn_code(x13, type = "13")
assert_any_are_isbn_codes(x13, type = "13")
# These checks should fail.
assertive.base::dont_stop(assert_all_are_isbn_codes(x10, type = "10"))
assertive.base::dont_stop(assert_all_are_isbn_codes(x13, type = "13"))
```

Index