Package ‘assertive.data.uk’

October 21, 2018

Type Package

Title Assertions to Check Properties of Strings

Version 0.0-2

Date 2018-10-21

Author Richard Cotton [aut, cre]

Maintainer Richard Cotton <richierocks@gmail.com>

Description A set of predicates and assertions for checking the properties of UK-specific 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.uk

BugReports https://bitbucket.org/richierocks/assertive.data.uk/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-uk.R' 'is-data-uk.R'

RoxygenNote 6.1.0

NeedsCompilation no

Repository CRAN

Date/Publication 2018-10-21 21:40:21 UTC

R topics documented:

`assert_all_are_uk_car_licences`

Is the string a valid UK car licence plate number?

Description

Checks that the input contains UK car licence plate numbers.

Usage

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

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

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

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

is_uk_car_licence(x)

is_uk_car_license(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_uk_national_insurance_number` returns `TRUE` if the input string contains a valid UK car licence plate number The assert_* function returns nothing but throw an error when the `is_*` function returns `FALSE`.

Note

A single space, in the appropriate place, is allowed but not compulsory.

References

Regex taken from http://www.regexlib.com/REDetails.aspx?regexp_id=527.

Examples

```
licences <- c(
  #1903 to 1931
  "A 1", "AA 9999",                   #ok
  "A 01",                             #zero prefix on number
  "S0", "G0", "RG0", "LM0",           #ok, special plates
  #1931 to 1963
  "AAA 1", "AAA 999",                 #ok
  "III 1", "QQQ 1", "ZZZ 1",          #disallowed letters
  "AAA 01",                           #zero prefix on number
  #1931 to 1963 alt
  "1 AAA", "9999 AAA",                #ok
  "1 III", "1 QQQ", "1 ZZZ",          #disallowed letters
  "01 AAA",                           #zero prefix on number
  #1963 to 1982
  "AAA 1A", "AAA 999A",               #ok
  "AAA 1I", "AAA 1O", "AAA 1Q",       #disallowed letters
  "AAA 1U", "AAA 1Z",
  "AAA 01A",                          #zero prefix on number
  #1982 to 2001
  "A1 AAA", "A999 AAA",               #ok
  "I1 AAA", "O1 AAA",                 #disallowed letters
  "U1 AAA", "Z1 AAA",
  "A01 AAA",                          #zero prefix on number
  #2001 to 2051
  "AA00 AAA", "AA99 AAA",             #ok
  "II00 AAA", "QQ00 AAA", "ZZ00 AAA", #disallowed letters
  "AA00 III", "AA00 QQQ"
)
is_uk_car_licence(licences)
assert_any_are_uk_car_licences(licences)
#These examples should fail.
assertive.base::dont_stop(assert_all_are_uk_car_licences(licences))
```

---

assert_all_are_uk_national_insurance_numbers

Is the string a valid UK national insurance number?

Description

Checks that the input contains UK national insurance numbers.

Usage

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

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

is_uk_national_insurance_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

`is_uk_national_insurance_number` returns `TRUE` if the input string contains a valid UK national insurance number. The assert_* function returns nothing but throw an error when the `is_*` function returns `FALSE`.

Note

A single space is allowed at the appropriate points (after the first two letters and after each pair of numbers) but not compulsory.

References

Regex taken from `http://www.regexlib.com/REDetails.aspx?regexp_id=527`.

Examples

```
ni_numbers <- c(
  "AA 00 00 00 A", "AA 00 00 00", "AA000000A",                #ok
  "ZZ 99 99 99 M", "ZZ 99 99 99", "ZZ999999M",
  "DA 00 00 00", "FA 00 00 00", "IA 00 00 00",                #bad first letter
  "QA 00 00 00", "UA 00 00 00", "VA 00 00 00",
  "AD 00 00 00", "AF 00 00 00", "AI 00 00 00", "AO 00 00 00", #bad second letter
  "AQ 00 00 00", "AU 00 00 00", "AV 00 00 00",
  "AA 00 00 00 E", "AA 00 00 00 G", "AA 00 00 00 H",          #bad final letter
  "AA 00 00 00 I", "AA 00 00 00 J", "AA 00 00 00 K",
  "AA 00 00 00 L", "AA 00 00 00 N", "AA 00 00 00 O",
  "AA 00 00 00 P", "AA 00 00 00 Q", "AA 00 00 00 R",
  "AA 00 00 00 S", "AA 00 00 00 T", "AA 00 00 00 U",
  "AA 00 00 00 V", "AA 00 00 00 W", "AA 00 00 00 X",
  "AA 00 00 00 Y", "AA 00 00 00 Z"
)
```

```
is_uk_national_insurance_number(ni_numbers)
assert_any_are_uk_national_insurance_numbers(ni_numbers)
#These examples should fail.
assertive.base::dont_stop(assert_all_are_uk_national_insurance_numbers(ni_numbers))
```

---

assert_all_are_uk_postcodes
Is the string a valid UK postcode?

Description

Checks that the input contains UK postcodes.

Usage

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

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

is_uk_postcode(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_uk_postcode 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

The function doesn't guarantee that the postcode actually exists. It should correctly return TRUE for genuine postcodes, and will weed out most badly formatted strings and non-existent areas, but some non-existent districts may incorrectly return TRUE. If you need 100 check against an up-to-date postcode database.

References

Regexes taken from https://en.wikipedia.org/wiki/Postcodes_in_the_United_Kingdom#Validation.

Examples

```
postcodes <- c(
  "SW1A 1AA", "SK11 9DW", "M34FP", "Le45ns", "TS25 2BZ", "gir 0aa"
)
is_uk_postcode(postcodes)
assert_all_are_uk_postcodes(postcodes)
```

`assert_all_are_uk_telephone_numbers`
Is the string a valid UK telephone number?

Description

Checks that the input contains UK telephone numbers.

Usage

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

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

is_uk_telephone_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

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

Note

The function doesn't guarantee that the phone number is in use, but checks that the format is correct, and that the area code exists. Spaces, hyphens and round brackets are allowed to appear in arbitrary places. The international UK prefix of 0044 or +44 is allowed.

References

The regex is adapted from one on the now defunct aa-asterisk.org.uk site with some additional consultation from https://en.wikipedia.org/wiki/List_of_United_Kingdom_dialling_codes

Examples

```
phone_nos <- c("+44 207 219 3475", "08457 90 90 90")
is_uk_telephone_number(phone_nos)
assert_all_are_uk_telephone_numbers(phone_nos)
```

Index