Package 'assertive.reflection'

July 31, 2020

Type Package

Title Assertions for Checking the State of R

Version 0.0-5

Date 2020-07-30

Author Richard Cotton [aut, cre]

Maintainer Richard Cotton <richierocks@gmail.com>

Description A set of predicates and assertions for checking the state and capabilities of R, the operating system it is running on, and the IDE being used. 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.reflection

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

Depends R (>= 3.0.0)

Imports assertive.base (>= 0.0-7), utils

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-32-64-bit.R' 'assert-is-current.R' 'assert-is-decimal-point.R' 'assert-is-ide.R' 'assert-is-on-os-path.R' 'assert-is-os.R' 'assert-is-r-mode.R' 'assert-is-r-version.R' 'assert-r-has-capability.R' 'is-32-64-bit.R' 'is-current.R' 'is-decimal-point.R' 'is-ide.R' 'is-on-os-path.R' 'is-os.R' 'is-r-mode.R' 'is-r-version.R' 'locale.R' 'r-has-capability.R'

RoxygenNote 7.1.1

Repository CRAN

NeedsCompilation no

Date/Publication 2020-07-31 01:00:14 UTC

R topics documented:

`assert_all_are_on_os_path` *Is the path on the OS path?*

Description

Is the specified path on the operating system search path?

Usage

```
assert_all_are_on_os_path(
  x,
  severity = getOption("assertive.severity", "stop")
)

assert_any_are_on_os_path(
  x,
  severity = getOption("assertive.severity", "stop")
)

is_on_os_path(x, .xname = get_name_in_parent(x))
```

Arguments

x	An path to check.
severity	How severe should the consequences of the assertion be? Either "stop", "warning", "message", or "none".
.xname	Not intended to be used directly.

Value

TRUE if the specified paths are on the OS search path.

Note

The OS search path is determined with Sys.getenv("path"). For files, the path of the containing folder is checked rather than the path of the file itself.

Examples

```
is_on_os_path(
  c(R.home("bin"), R.home("etc"), "a nonexistent path")
) # probably c(TRUE, FALSE, FALSE)
```

---

assert_is_64_bit_os *What OS is running?*

Description

Is the operating system in this machine Windows/Unix/Mac based.

Usage

```
assert_is_64_bit_os(severity = getOption("assertive.severity", "stop"))

assert_is_32_bit(severity = getOption("assertive.severity", "stop"))

assert_is_64_bit(severity = getOption("assertive.severity", "stop"))

assert_is_bsd(severity = getOption("assertive.severity", "stop"))

assert_is_linux(severity = getOption("assertive.severity", "stop"))

assert_is_mac(severity = getOption("assertive.severity", "stop"))

assert_is_osx(severity = getOption("assertive.severity", "stop"))

assert_is_osx_cheetah(severity = getOption("assertive.severity", "stop"))

assert_is_osx_puma(severity = getOption("assertive.severity", "stop"))
```

```
assert_is_osx_jaguar(severity = getOption("assertive.severity", "stop"))
assert_is_osx_panther(severity = getOption("assertive.severity", "stop"))
assert_is_osx_tiger(severity = getOption("assertive.severity", "stop"))
assert_is_osx_leopard(severity = getOption("assertive.severity", "stop"))
assert_is_osx_snow_leopard(severity = getOption("assertive.severity", "stop"))
assert_is_osx_lion(severity = getOption("assertive.severity", "stop"))
assert_is_osx_mountain_lion(severity = getOption("assertive.severity", "stop"))
assert_is_osx_mavericks(severity = getOption("assertive.severity", "stop"))
assert_is_osx_yosemite(severity = getOption("assertive.severity", "stop"))
assert_is_osx_el_capitan(severity = getOption("assertive.severity", "stop"))
assert_is_macos_sierra(severity = getOption("assertive.severity", "stop"))
assert_is_macos_high_sierra(severity = getOption("assertive.severity", "stop"))
assert_is_macos_mojave(severity = getOption("assertive.severity", "stop"))
assert_is_macos_catalina(severity = getOption("assertive.severity", "stop"))
assert_is_macos_big_sur(severity = getOption("assertive.severity", "stop"))
assert_is_solaris(severity = getOption("assertive.severity", "stop"))
assert_is_unix(severity = getOption("assertive.severity", "stop"))
assert_is_windows(severity = getOption("assertive.severity", "stop"))
assert_is_windows_vista(severity = getOption("assertive.severity", "stop"))
assert_is_windows_7(severity = getOption("assertive.severity", "stop"))
assert_is_windows_8(severity = getOption("assertive.severity", "stop"))
assert_is_windows_8.1(severity = getOption("assertive.severity", "stop"))
assert_is_windows_10(severity = getOption("assertive.severity", "stop"))
assert_is_windows_server_2008(
```

```
  severity = getOption("assertive.severity", "stop")
)

assert_is_windows_server_2008_r2(
  severity = getOption("assertive.severity", "stop")
)

assert_is_windows_server_2012(
  severity = getOption("assertive.severity", "stop")
)

assert_is_windows_server_2012_r2(
  severity = getOption("assertive.severity", "stop")
)

is_64_bit_os()

is_32_bit()

is_64_bit()

is_bsd()

is_linux()

is_mac()

is_osx()

is_osx_cheetah()

is_osx_puma()

is_osx_jaguar()

is_osx_panther()

is_osx_tiger()

is_osx_leopard()

is_osx_snow_leopard()

is_osx_lion()

is_osx_mountain_lion()

is_osx_mavericks()
```

```
is_osx_yosemite()

is_osx_el_capitan()

is_macos_sierra()

is_macos_high_sierra()

is_macos_mojave()

is_macos_catalina()

is_macos_big_sur()

is_solaris()

is_unix()

is_windows()

is_windows_vista()

is_windows_7()

is_windows_8()

is_windows_8.1()

is_windows_10()

is_windows_server_2008()

is_windows_server_2008_r2()

is_windows_server_2012()

is_windows_server_2012_r2()

is_windows_server_2016()

is_windows_server_2019()
```

Arguments

`severity`	How severe should the consequences of the assertion be? Either `"stop"`, `"warning"`, `"message"`, or `"none"`.

Value

is_windows returns TRUE if the OS on the current platform is Microsoft windows-based. is_unix returns TRUE if the OS is Unix based (pretty much anything that isn't Windows, including OS X). is_mac, is_linux, is_bsd, is_solaris return TRUE if the OS is Apple OS X, Linux, FreeBSD/NetBSD, or Solaris respectively. is_64_bit_os returns TRUE when the operating system is 64-bit. The assert_* functions return nothing but throw an error if the corresponding is_* functions return FALSE.

References

With the exception of is_windows and is_unix that use .Platform$OS.type, the OS is determined from Sys.info()[["sysname"]], which (not on Windows) is calculated via the OS uname program. GNU has more information on the return value: https://www.gnu.org/software/libc/manual/html_node/Platform-Type.html and Wikipedia has a nice list of possible values: https://en.wikipedia.org/wiki/Uname#Examples The names for different versions of Windows are decribed in: http://svn.r-project.org/R/trunk/src/library/utils/src/windows/util.c

See Also

.Platform, Sys.info, version, and win.version.

Examples

```
is_unix()
is_linux()
is_bsd()
is_solaris()
if(is_windows())
{
  assertive.base::dont_stop({
    assert_is_windows_vista()
    assert_is_windows_7()
    assert_is_windows_8()
    assert_is_windows_8.1()
    assert_is_windows_10()
    assert_is_windows_server_2008()
    assert_is_windows_server_2008_r2()
    assert_is_windows_server_2012()
    assert_is_windows_server_2012_r2()
  })
}
if(is_osx()) # is_mac is a synonym
{
  assertive.base::dont_stop({
    assert_is_osx_cheetah()
    assert_is_osx_puma()
    assert_is_osx_jaguar()
    assert_is_osx_panther()
    assert_is_osx_tiger()
    assert_is_osx_leopard()
```

```
    assert_is_osx_snow_leopard()
    assert_is_osx_lion()
    assert_is_osx_mountain_lion()
    assert_is_osx_mavericks()
    assert_is_osx_yosemite()
    assert_is_osx_el_capitan()
    assert_is_macos_sierra() # note the change from OSX to macOS
  })
}
is_32_bit()
is_64_bit()
assertive.base::dont_stop(assert_is_windows())
assertive.base::dont_stop(assert_is_unix())
```

`assert_is_architect` *Are you running R?*

Description

Checks to see what type of R you are running.

Usage

```
assert_is_architect(severity = getOption("assertive.severity", "stop"))

assert_is_emacs(severity = getOption("assertive.severity", "stop"))

assert_is_revo_r(severity = getOption("assertive.severity", "stop"))

assert_is_rstudio(severity = getOption("assertive.severity", "stop"))

assert_is_rstudio_desktop(severity = getOption("assertive.severity", "stop"))

assert_is_rstudio_server(severity = getOption("assertive.severity", "stop"))

assert_is_visual_studio(severity = getOption("assertive.severity", "stop"))

assert_is_r(severity = getOption("assertive.severity", "stop"))

assert_is_r_alpha(severity = getOption("assertive.severity", "stop"))

assert_is_r_beta(severity = getOption("assertive.severity", "stop"))

assert_is_r_devel(severity = getOption("assertive.severity", "stop"))

assert_is_r_patched(severity = getOption("assertive.severity", "stop"))

assert_is_r_release_candidate(
```

```
  severity = getOption("assertive.severity", "stop")
)

assert_is_r_release(severity = getOption("assertive.severity", "stop"))

assert_is_r_revised(severity = getOption("assertive.severity", "stop"))

assert_is_r_stable(severity = getOption("assertive.severity", "stop"))

is_architect()

is_emacs()

is_revo_r()

is_rstudio()

is_visual_studio()

is_r()

is_r_alpha()

is_r_beta()

is_r_devel()

is_r_patched()

is_r_release_candidate()

is_r_release()

is_r_revised()

is_r_stable()
```

Arguments

severity	How severe should the consequences of the assertion be? Either "stop", "warning", "message", or "none".

Value

is_r wraps is.R, providing more information on failure. is_r_stable, is_r_patched, is_r_devel, etc., tell you what type of R build you are running. is_architect, is_rstudio and is_revo_r tell you if you are running Architect/StatET, RStudio, or Revolution Analytics' Revolution R build. is_slave_r tells you if you are running a slave instance of R (e.g. when building a package with

devtools or using a cluster). The assert_* functions return nothing but throw an error if the corresponding is_* function returns FALSE.

References

http://www.revolutionanalytics.com/revolution-r-open

See Also

is.R, version, isAvailable.

Examples

```
# If this is FALSE, you really need to ditch that old copy of S-PLUS
is_r()
assertive.base::dont_stop(assert_is_r())
# Release, patched, devel, etc.
is_r_release()
is_r_patched()
is_r_devel()
is_r_alpha()
is_r_beta()
is_r_release_candidate()
is_r_revised()
switch(
  version$status,
  Patched                       = assert_is_r_patched(),
  "Under development (unstable)" = assert_is_r_devel(),
  alpha                         = assert_is_r_alpha(),
  beta                          = assert_is_r_beta(),
  RC                            = assert_is_r_release_candidate(),
  Revised                       = assert_is_r_revised(),
  assert_is_r_release()
)
# IDE
is_architect()
is_emacs()
is_visual_studio()
is_rstudio()
# Custom R distribution
is_revo_r()
```

---

assert_is_batch_mode *How is R running?*

Description

Tests to see if R is running in batch mode/interactively.

Usage

```
assert_is_batch_mode(severity = getOption("assertive.severity", "stop"))

assert_is_interactive(severity = getOption("assertive.severity", "stop"))

assert_is_r_slave(severity = getOption("assertive.severity", "stop"))

assert_is_slave_r(severity = getOption("assertive.severity", "stop"))

is_batch_mode()

is_interactive()

is_r_slave()

is_slave_r()
```

Arguments

severity	How severe should the consequences of the assertion be? Either "stop", "warning", "message", or "none".

Value

is_batch_mode returns TRUE if R is running in batch mode. is_interactive returns TRUE if R is running interactively.

See Also

EnvVar and interactive.

Examples

```
is_batch_mode()
is_interactive()
is_r_slave()
```

---

assert_is_comma_for_decimal_point

What does the current locale specify for the decimal point?

Description

Does the current locale specify a comma or a period for the decimal point?

Usage

```
assert_is_comma_for_decimal_point(
  severity = getOption("assertive.severity", "stop")
)

assert_is_period_for_decimal_point(
  severity = getOption("assertive.severity", "stop")
)

is_xxx_for_decimal_point(dp, type = c("numbers", "money"))

is_comma_for_decimal_point(type = c("numbers", "money"))

is_period_for_decimal_point(type = c("numbers", "money"))
```

Arguments

severity	How severe should the consequences of the assertion be? Either "stop", "warning", "message", or "none".
dp	Character to be used as a decimal point.
type	Decimal point for numbers or money?

Value

is_comma_for_decimal_point returns TRUE when the current locale uses a comma for a decimal place, as determined by Sys.localeconv. Similarly, is_period_for_decimal_point returns TRUE when the current locale uses a period (a.k.a. full stop) for a decimal place. If R has been compiled without support for locales, then the value will always be NA.

References

http://www.cplusplus.com/reference/clocale/lconv/

See Also

Sys.localeconv

Examples

```
# Current settings:
is_comma_for_decimal_point()
is_comma_for_decimal_point("money")
# Or equivalently:
is_period_for_decimal_point()
is_period_for_decimal_point("money")
# A useful guess for reading in files:
read_csv <- if(is_comma_for_decimal_point()) read.csv else read.csv2
## Not run:
# Force locale and test (may require admin rights)
```

```
current_locale <- sys_get_locale()
a_period_locale <- if(is_windows())
{
  "English_United Kingdom.1252"
} else if(is_mac())
{
  "en_GB"
} else if(is_linux())
{
  "en_GB.utf8"
} else
{
  "en"
}
sys_set_locale(LC_ALL = a_period_locale)
assert_is_period_for_decimal_point()
a_comma_locale <- if(is_windows())
{
  "French_France.1252"
} else if(is_mac())
{
  "fr_FR"
} else if(is_linux())
{
  "fr_FR.utf8"
} else
{
  "fr"
}
sys_set_locale(LC_ALL = a_comma_locale)
assert_is_comma_for_decimal_point()
suppressWarnings(sys_set_locale(l = current_locale))

## End(Not run)
```

`assert_is_package_current` *Is the installed version of a package current?*

Description

Checks to see if the installed version of a package is current.

Usage

```
assert_is_package_current(...)

assert_all_are_current_packages(
  x,
  lib.loc = .libPaths(),
```

```
  repos = getOption("repos"),
  type = getOption("pkgType"),
  severity = getOption("assertive.severity", "stop")
)

assert_any_are_current_packages(
  x,
  lib.loc = .libPaths(),
  repos = getOption("repos"),
  type = getOption("pkgType"),
  severity = getOption("assertive.severity", "stop")
)

is_package_current(
  x = NULL,
  lib.loc = .libPaths(),
  repos = getOption("repos"),
  type = getOption("pkgType"),
  .xname = get_name_in_parent(x)
)
```

Arguments

`...`	Passed to and from deprecated `assert_is_current_package`.
`x`	A character vector of package names, or `NULL` to check all installed packages.
`lib.loc`	A character vector of paths to local package libraries.
`repos`	A character vector of URLs to repositories to check for new package versions.
`type`	Check the repository for source or binary packages?
`severity`	How severe should the consequences of the assertion be? Either `"stop"`, `"warning"`, `"message"`, or `"none"`.
`.xname`	Not intended to be used directly.

Value

`is_package_current` returns a logical vector that is `TRUE` whenever the package version matches the one in the repository. `NA` is returned for non-installed packages. The `assert_*` functions throw an error in the event of failure.

See Also

`old.packages`, on which this is based, which has advanced usage features.

Examples

```
# This test is marked "dont-test" since it involves a connection to
# repositories which is potentially long running.
is_package_current(c("assertive.base", "assertive.reflection", "NONEXISTENTPKG"))
```

assert_is_rstudio_current *Is RStudio the current version?*

Description

Checks to see if the running version of RStudio is the current version.

Usage

```
assert_is_rstudio_current(severity = getOption("assertive.severity", "stop"))

is_rstudio_current()
```

Arguments

severity	How severe should the consequences of the assertion be? Either "stop", "warning", "message", or "none".

Value

is_rstudio_current returns TRUE or FALSE, and assert_is_rstudio_current throws an error in the event of an out of date RStudio. Non-RStudio IDEs throw an error.

References

This function is engineered from the downloadUpdateInfo function from https://github.com/rstudio/rstudio/blob/master/src/cpp/session/modules/SessionUpdates.R where the string for the OS is described in beginUpdateCheck from https://github.com/rstudio/rstudio/blob/master/src/cpp/session/modules/SessionUpdates.cpp

See Also

is_rstudio, is_rstudio_desktop

assert_is_r_current *Is this version of R up to date?*

Description

Check if this version of R is as new as the current release version of R.

Usage

```
assert_is_r_current(severity = getOption("assertive.severity", "stop"))

assert_is_current_r(severity = getOption("assertive.severity", "stop"))

is_r_current(
  cran = getOption("repos", c(CRAN = "http://cran.r-project.org"))["CRAN"]
)
```

Arguments

severity	How severe should the consequences of the assertion be? Either "stop", "warning", "message", or "none".
cran	A string giving the URL of the CRAN repository to check.

Value

An object of class R_system_version giving the current release version of R.

Note

Development versions of R can have versions higher than the current release version of R. For convenience, these will return TRUE.

Examples

```
# This example is marked "don't test" since it requires an
# internet connection and is potentially long running
is_r_current()
```

---

assert_r_can_find_tools

Can R find tools?

Description

Checks to see if R can see command line tools.

Usage

```
assert_r_can_find_tools(
  tools,
  severity = getOption("assertive.severity", "stop")
)
```

```
assert_r_can_compile_code(severity = getOption("assertive.severity", "stop"))

assert_r_can_build_translations(
  severity = getOption("assertive.severity", "stop")
)

assert_r_can_find_java(
  java_type = c("same_as_r", "any", "64bit", "32bit"),
  severity = getOption("assertive.severity", "stop")
)

r_can_find_tools(tools)

r_can_compile_code()

r_can_build_translations()

r_can_find_java(java_type = c("same_as_r", "any", "64bit", "32bit"))
```

Arguments

tools	A character vector of tools to look for.
severity	How severe should the consequences of the assertion be? Either "stop", "warning", "message", or "none".
java_type	A string denoting the type of Java to look for (either 32 or 64 bit).

Value

The is_* functions return TRUE if the input is within an interval. The assert_* functions return nothing but throw an error if the corresponding is_* function returns FALSE.

Note

r_can_compile_code is a convenience function looking for gcc and make.

r_can_build_translations is a convenience function looking for gettext and msgfmt.

See Also

Sys.which

Examples

```
r_can_find_tools(c("latex", "pdflatex"))
r_can_compile_code()
r_can_build_translations()
r_can_find_java()
assertive.base::dont_stop({
  assert_r_can_find_tools(c("latex", "pdflatex"))
  assert_r_can_compile_code()
```

```
  assert_r_can_build_translations()
  assert_r_can_find_java("64bit")
})
```

`assert_r_has_jpeg_capability` *Does R have a capability?*

Description

Check to see if R has a specific capability.

Usage

```
assert_r_has_jpeg_capability(
  severity = getOption("assertive.severity", "stop")
)

assert_r_has_png_capability(severity = getOption("assertive.severity", "stop"))

assert_r_has_tiff_capability(
  severity = getOption("assertive.severity", "stop")
)

assert_r_has_tcltk_capability(
  severity = getOption("assertive.severity", "stop")
)

assert_r_has_x11_capability(severity = getOption("assertive.severity", "stop"))

assert_r_has_aqua_capability(
  severity = getOption("assertive.severity", "stop")
)

assert_r_has_http_ftp_capability(
  severity = getOption("assertive.severity", "stop")
)

assert_r_has_sockets_capability(
  severity = getOption("assertive.severity", "stop")
)

assert_r_has_libxml_capability(
  severity = getOption("assertive.severity", "stop")
)

assert_r_has_fifo_capability(
```

```
  severity = getOption("assertive.severity", "stop")
)

assert_r_has_cledit_capability(
  severity = getOption("assertive.severity", "stop")
)

assert_r_has_iconv_capability(
  severity = getOption("assertive.severity", "stop")
)

assert_r_has_nls_capability(severity = getOption("assertive.severity", "stop"))

assert_r_has_rprof_capability(
  severity = getOption("assertive.severity", "stop")
)

assert_r_has_profmem_capability(
  severity = getOption("assertive.severity", "stop")
)

assert_r_has_cairo_capability(
  severity = getOption("assertive.severity", "stop")
)

assert_r_has_icu_capability(severity = getOption("assertive.severity", "stop"))

assert_r_has_long_double_capability(
  severity = getOption("assertive.severity", "stop")
)

assert_r_has_libcurl_capability(
  severity = getOption("assertive.severity", "stop")
)

r_has_jpeg_capability()

r_has_png_capability()

r_has_tiff_capability()

r_has_tcltk_capability()

r_has_x11_capability()

r_has_aqua_capability()

r_has_http_ftp_capability()
```

```
r_has_sockets_capability()

r_has_libxml_capability()

r_has_fifo_capability()

r_has_cledit_capability()

r_has_iconv_capability()

r_has_nls_capability()

r_has_rprof_capability()

r_has_profmem_capability()

r_has_cairo_capability()

r_has_icu_capability()

r_has_long_double_capability()

r_has_libcurl_capability()
```

Arguments

severity	How severe should the consequences of the assertion be? Either "stop", "warning", "message", or "none".

Value

The is_* functions return TRUE if R has the capability and FALSE (with a cause) otherwise. The assert_* functions return nothing but throw an error if the corresponding is_* function returns FALSE.

See Also

capabilities

Examples

```
## Not run:
if(r_has_png_capability())
{
  png("test.png")
  with(cars, plot(speed, dist))
  dev.off()
} else
{
```

```
  pdf("test.pdf")
  with(cars, plot(speed, dist))
  dev.off()
}

## End(Not run)
```

`is_rstudio_desktop` *Is RStudio running in desktop or server mode?*

Description

Checks for RStudio desktop or server version.

Usage

```
is_rstudio_desktop()

is_rstudio_server()
```

References

The values that RStudio uses for its mode are defined in https://github.com/rstudio/rstudio/blob/master/src/cpp/session/include/session/SessionConstants.hpp via the constants kSessionProgramModeDe and kSessionProgramModeServer.

See Also

is_rstudio, is_rstudio_current

Examples

```
is_rstudio_desktop()
is_rstudio_server()
```

`rstudio_version_info` *Get RStudio's version information*

Description

Wrapper to .rs.api.versionInfo.

Usage

```
rstudio_version_info()
```

sys_get_locale *Get or set the system locale*

Description

Wrappers to `Sys.getlocale` and `Sys.setlocale` for getting and setting the system locale.

Usage

```
sys_get_locale(simplify = FALSE, remove_empty_categories = TRUE)

sys_set_locale(..., l = list())
```

Arguments

`simplify`	If `TRUE`, the locale settings are returned as a character vector, otherwise a list.
`remove_empty_categories`	if `TRUE`, don't include empty categories.
`...`	Name-value pairs of locale categories to set.
`l`	A list, as an alternative method of passing local categories to set.

Value

A named list or vector giving the system locale names. `sys_set_locale` invisibly returns the locale settings *before* making changes (like `setwd` and `options` do).

See Also

`Sys.getlocale`.

Examples

```
(current_locale <- sys_get_locale())

# Output simplified to character vector
sys_get_locale(simplify = TRUE)
## Not run:
# Not run since it (temporarily) affects system settings
english <- if(is_windows()) "English.United_Kingdom" else
  if(is_mac()) "en_GB" else
  if(is_linux()) "en_GB.utf8" else
  "en"
sys_set_locale(LC_MONETARY = english)
sys_get_locale()
sys_set_locale(l = current_locale)  #restore everything

## End(Not run)
```

Index