Package 'sessioncheck'

September 13, 2026

Title Checks Session Status

Version 0.2.0

Description Provides tools for checking whether an R session is in a clean state, including the global environment, attached packages, loaded namespaces, attached environments, session run time, R options, locale settings, and system environment variables. Intended as a safer replacement for the common 'rm(list = ls())' idiom: rather than silently wiping the global environment, sessioncheck() surfaces problems so the user can make an informed decision. The package also supplies tools for documenting the session state, to aid in the overall process.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

URL https://github.com/djnavarro/sessioncheck, https://sessioncheck.djnavarro.net/

BugReports https://github.com/djnavarro/sessioncheck/issues

Suggests knitr, rmarkdown, sessioninfo, spelling, testthat (>= 3.0.0)

Config/testthat/edition 3

Config/Needs/website djnavarro/waeponwifestre

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Danielle Navarro [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-7648-6578>), Meghan Harris [ctb] (ORCID: <https://orcid.org/0000-0003-3922-8101>)

Maintainer Danielle Navarro <djnavarro@protonmail.com>

Repository CRAN

Date/Publication 2026-09-13 11:40:02 UTC

Contents

`check_attached_environments` *Check environments attached to the search path*

Description

Individual session check function that inspects the names of attached non-package environments. Session checkers can produce errors, warnings, or messages if requested.

Usage

```
check_attached_environments(
  action = "warn",
  allow_attached_environments = NULL,
  action_on_pass = "none"
)
```

Arguments

`action`	Behavior to take if the status is not clean. Possible values are "error", "warn", "message", and "none". The default is `action = "warn"`.
`allow_attached_environments`	Character vector containing names of environments that are "allowed", and will not trigger an action if attached to the search path. The default is `allow_attached_environments = NULL`, in which case package environments and a set of known IDE-injected environments (e.g. "tools:rstudio") are allowed.
`action_on_pass`	Behavior to take if the status is clean. Possible values are "message" and "none". The default is `action_on_pass = "none"`.

Details

This checker inspects all environments on the search path. This includes attached packages, anything added using `attach()`, and the global environment. When `allow_attached_environments = NULL`, package environments do not trigger an action, nor do "tools:rstudio", "tools:positron", "tools:callr", or "Autoloads". The global environment and the package environment for the **base** package never trigger actions.

Value

Invisibly returns an object of class `sessioncheck_status`.

See Also

`check_attached_packages()`, `check_loaded_namespaces()`, `check_globalenv_objects()`, `check_sessiontime()`, `check_required_options()`, `check_required_locale()`, `check_required_sysenv()`, `check_working_directory()`

Examples

```
check_attached_environments(action = "message")

# a session with no unexpected environments attached: `action` only
# controls what happens when a problem *is* found, so use
# `action_on_pass = "message"` to confirm the clean result instead
check_attached_environments(
  action = "none",
  allow_attached_environments = search(),
  action_on_pass = "message"
)
```

---

check_attached_packages

Check attached packages

Description

Individual session check function that inspects the attached packages. Session checkers can produce errors, warnings, or messages if requested.

Usage

```
check_attached_packages(
  action = "warn",
  allow_attached_packages = NULL,
  action_on_pass = "none"
)
```

Arguments

`action`	Behavior to take if the status is not clean. Possible values are "error", "warn", "message", and "none". The default is `action = "warn"`.
`allow_attached_packages`	Character vector containing names of packages that are "allowed", and will not trigger an action if attached to the search path. The default is `allow_attached_packages = NULL`, in which case only base-priority packages are allowed.
`action_on_pass`	Behavior to take if the status is clean. Possible values are "message" and "none". The default is `action_on_pass = "none"`.

Details

This checker inspects the list of packages that have been attached to the search path (e.g., via `library()`). Regardless of the value of `allow_attached_packages`, R packages that have "base" priority (e.g., **base**, **utils**, and **grDevices**) do not trigger an action. When `allow_attached_packages = NULL` these are the only packages that will not trigger actions.

Value

Invisibly returns an object of class `sessioncheck_status`.

See Also

`check_loaded_namespaces()`, `check_globalenv_objects()`, `check_attached_environments()`, `check_sessiontime()`, `check_required_options()`, `check_required_locale()`, `check_required_sysenv()`, `check_working_directory()`

Examples

```
check_attached_packages(action = "message")

# a session with no unexpected packages attached: `action` only controls
# what happens when a problem *is* found, so use `action_on_pass = "message"`
# to confirm the clean result instead
check_attached_packages(
  action = "none",
  allow_attached_packages = .packages(),
  action_on_pass = "message"
)
```

---

check_globalenv_objects

Check global environment objects

Description

Individual session check function that inspects the contents of the global environment. Session checkers can produce errors, warnings, or messages if requested.

Usage

```
check_globalenv_objects(
  action = "warn",
  allow_globalenv_objects = NULL,
  action_on_pass = "none"
)
```

Arguments

action	Behavior to take if the status is not clean. Possible values are "error", "warn", "message", and "none". The default is action = "warn".
allow_globalenv_objects	Character vector containing names of objects that are "allowed", and will not trigger an action. The default is allow_globalenv_objects = NULL, in which case dot-prefixed objects (e.g. .Random.seed) are allowed.
action_on_pass	Behavior to take if the status is clean. Possible values are "message" and "none". The default is action_on_pass = "none".

Details

This checker inspects the state of the global environment and takes action based on the objects found there. When allow_globalenv_objects = NULL, variables in the global environment will not trigger an action if the name starts with a dot. For example, .Random.seed and .Last.value do not trigger actions by default.

Value

Invisibly returns an object of class sessioncheck_status.

See Also

check_attached_packages(), check_loaded_namespaces(), check_attached_environments(), check_sessiontime(), check_required_options(), check_required_locale(), check_required_sysenv(), check_working_directory()

Examples

```
check_globalenv_objects(action = "message")

# a session with no unexpected objects in the global environment:
# `action` only controls what happens when a problem *is* found, so use
# `action_on_pass = "message"` to confirm the clean result instead
check_globalenv_objects(
  action = "none",
```

```
  allow_globalenv_objects = ls(envir = .GlobalEnv, all.names = TRUE),
  action_on_pass = "message"
)
```

`check_loaded_namespaces` *Check loaded namespaces*

Description

Individual session check function that inspects the loaded namespaces. Session checkers can produce errors, warnings, or messages if requested.

Usage

```
check_loaded_namespaces(
  action = "warn",
  allow_loaded_namespaces = NULL,
  action_on_pass = "none"
)
```

Arguments

`action`	Behavior to take if the status is not clean. Possible values are "error", "warn", "message", and "none". The default is `action = "warn"`.
`allow_loaded_namespaces`	Character vector containing names of packages that are "allowed", and will not trigger an action if loaded via namespace. The default is `allow_loaded_namespaces = NULL`, in which case only base-priority packages and the **sessioncheck** namespace itself are allowed.
`action_on_pass`	Behavior to take if the status is clean. Possible values are "message" and "none". The default is `action_on_pass = "none"`.

Details

This checker inspects the list of loaded namespaces (packages that have been loaded but not attached). Regardless of the value of `allow_loaded_namespaces`, R packages that have "base" priority (e.g., **base**, **utils**, and **grDevices**) do not trigger an action, nor does the **sessioncheck** package itself, since the package namespace must be loaded in order to call the function.

Value

Invisibly returns an object of class `sessioncheck_status`.

See Also

check_attached_packages(), check_globalenv_objects(), check_attached_environments(), check_sessiontime(), check_required_options(), check_required_locale(), check_required_sysenv(), check_working_directory()

Examples

```
check_loaded_namespaces(action = "message")

# a session with no unexpected namespaces loaded: `action` only controls
# what happens when a problem *is* found, so use `action_on_pass = "message"`
# to confirm the clean result instead
check_loaded_namespaces(
  action = "none",
  allow_loaded_namespaces = loadedNamespaces(),
  action_on_pass = "message"
)
```

---

check_required_locale *Check required values for locale settings*

Description

Individual session check function that inspects the locale settings. Session checkers can produce errors, warnings, or messages if requested.

Usage

```
check_required_locale(
  action = "warn",
  required_locale = NULL,
  action_on_pass = "none"
)
```

Arguments

action	Behavior to take if the status is not clean. Possible values are "error", "warn", "message", and "none". The default is action = "warn".
required_locale	A named list of required locale settings. If any of these are missing or have different values to the required values, an action is triggered. The default is required_locale = NULL, which means there is nothing to compare against, so the check always passes.
action_on_pass	Behavior to take if the status is clean. Possible values are "message" and "none". The default is action_on_pass = "none".

Value

Invisibly returns an object of class `sessioncheck_status`.

See Also

check_attached_packages(), check_loaded_namespaces(), check_globalenv_objects(), check_attached_enviro
check_sessiontime(), check_required_options(), check_required_sysenv(), check_working_directory()

Examples

```
check_required_locale(action = "message", required_locale = list(LC_TIME = "en_US.UTF-8"))

# a required locale setting that is present, but has a different value:
# reports the expected and actual values
check_required_locale(action = "message", required_locale = list(LC_CTYPE = "not-a-real-locale"))

# a required locale setting that isn't part of the current locale at
# all: reported as missing, rather than lumped in with the
# mismatched-value case above
check_required_locale(action = "message", required_locale = list(LC_MADEUP = "en_US.UTF-8"))

# a required locale setting matching its current value: `action` only
# controls what happens when a problem *is* found, so use
# `action_on_pass = "message"` to confirm the clean result instead
check_required_locale(
  action = "none",
  required_locale = list(LC_COLLATE = Sys.getlocale("LC_COLLATE")),
  action_on_pass = "message"
)
```

---

check_required_options

Check required values for options

Description

Individual session check function that inspects the options. Session checkers can produce errors, warnings, or messages if requested.

Usage

```
check_required_options(
  action = "warn",
  required_options = NULL,
  action_on_pass = "none"
)
```

Arguments

action	Behavior to take if the status is not clean. Possible values are "error", "warn", "message", and "none". The default is action = "warn".
required_options	A named list of required options. If any of these options are missing or have different values to the required values, an action is triggered. The default is required_options = NULL, which means there is nothing to compare against, so the check always passes.
action_on_pass	Behavior to take if the status is clean. Possible values are "message" and "none". The default is action_on_pass = "none".

Value

Invisibly returns an object of class sessioncheck_status.

See Also

check_attached_packages(), check_loaded_namespaces(), check_globalenv_objects(), check_attached_enviro
check_sessiontime(), check_required_locale(), check_required_sysenv(), check_working_directory()

Examples

```
check_required_options(action = "message", required_options = list(scipen = 0L, max.print = 50L))

# a required option that is present, but has a different value: reports
# the expected and actual values
old <- options(scipen = 0L)
check_required_options(action = "message", required_options = list(scipen = 100L))
options(old)

# a required option that is not set at all: reported as missing, rather
# than lumped in with the mismatched-value case above
check_required_options(
  action = "message",
  required_options = list(a_totally_unset_option = TRUE)
)

# a long vector value is summarized rather than printed in full
old <- options(scipen = 0L)
check_required_options(action = "message", required_options = list(scipen = 1:5000))
options(old)

# non-atomic values (e.g. lists) are described by role -- "user-supplied"
# vs. "a different" -- rather than by content, since two unequal lists
# would otherwise both print as the uninformative "<list>"
old <- options(sessioncheck_example = list(a = 1))
check_required_options(
  action = "message",
  required_options = list(sessioncheck_example = list(b = 2))
)
```

```
options(old)

# a required option matching its current value: `action` only controls
# what happens when a problem *is* found, so use
# `action_on_pass = "message"` to confirm the clean result instead
check_required_options(
  action = "none",
  required_options = list(digits = getOption("digits")),
  action_on_pass = "message"
)
```

`check_required_sysenv` *Check required values for system environment variables*

Description

Individual session check function that inspects system environment variables. Session checkers can produce errors, warnings, or messages if requested.

Usage

```
check_required_sysenv(
  action = "warn",
  required_sysenv = NULL,
  action_on_pass = "none"
)
```

Arguments

`action`	Behavior to take if the status is not clean. Possible values are "error", "warn", "message", and "none". The default is `action = "warn"`.
`required_sysenv`	A named list of required system environment variables. If any of these variables are missing or have different values to the required values, an action is triggered. The default is `required_sysenv = NULL`, which means there is nothing to compare against, so the check always passes.
`action_on_pass`	Behavior to take if the status is clean. Possible values are "message" and "none". The default is `action_on_pass = "none"`.

Value

Invisibly returns an object of class `sessioncheck_status`.

See Also

`check_attached_packages()`, `check_loaded_namespaces()`, `check_globalenv_objects()`, `check_attached_enviro`
`check_sessiontime()`, `check_required_options()`, `check_required_locale()`, `check_working_directory()`

Examples

```
check_required_sysenv(action = "message", required_sysenv = list(R_TEST = "value"))

# a required variable that is present, but has a different value: reports
# the expected and actual values (R_HOME is set by R itself, so this is
# reliably a mismatch rather than a missing variable)
check_required_sysenv(action = "message", required_sysenv = list(R_HOME = "not-the-real-path"))

# a required variable that is not set at all: reported as missing,
# rather than lumped in with the mismatched-value case above
check_required_sysenv(
  action = "message",
  required_sysenv = list(SESSIONCHECK_EXAMPLE_UNSET_VAR = "value")
)

# a required variable matching its current value: `action` only
# controls what happens when a problem *is* found, so use
# `action_on_pass = "message"` to confirm the clean result instead
check_required_sysenv(
  action = "none",
  required_sysenv = list(R_HOME = Sys.getenv("R_HOME")),
  action_on_pass = "message"
)
```

---

check_sessiontime *Check session run time*

Description

Individual session check function that inspects the session run time information. Session checkers can produce errors, warnings, or messages if requested.

Usage

```
check_sessiontime(
  action = "warn",
  max_sessiontime = NULL,
  action_on_pass = "none"
)
```

Arguments

action	Behavior to take if the status is not clean. Possible values are "error", "warn", "message", and "none". The default is `action = "warn"`.
max_sessiontime	Maximum session time permitted in seconds before the checker takes action. The default is `max_sessiontime = 300`.
action_on_pass	Behavior to take if the status is clean. Possible values are "message" and "none". The default is `action_on_pass = "none"`.

Value

Invisibly returns an object of class `sessioncheck_status`.

See Also

`check_attached_packages()`, `check_loaded_namespaces()`, `check_globalenv_objects()`, `check_attached_enviro`
`check_required_options()`, `check_required_locale()`, `check_required_sysenv()`, `check_working_directory()`

Examples

```
check_sessiontime(action = "message")

# a session that has run past the threshold: reports the elapsed time
# and the threshold together, both in human-readable units
check_sessiontime(action = "message", max_sessiontime = 0)

# a session comfortably within the threshold: `action` only controls
# what happens when a problem *is* found, so use
# `action_on_pass = "message"` to confirm the clean result instead
check_sessiontime(action = "none", max_sessiontime = Inf, action_on_pass = "message")
```

---

check_working_directory

Check the working directory

Description

Individual session check function that inspects the current working directory. Session checkers can produce errors, warnings, or messages if requested.

Usage

```
check_working_directory(
  action = "warn",
  required_wd = NULL,
  action_on_pass = "none"
)
```

Arguments

`action`	Behavior to take if the status is not clean. Possible values are "error", "warn", "message", and "none". The default is `action = "warn"`.
`required_wd`	A single character path giving the working directory the session is expected to be in. If any other directory is currently in use, an action is triggered. The default is `required_wd = NULL`, which means there is nothing to compare against, so the check always passes.
`action_on_pass`	Behavior to take if the status is clean. Possible values are "message" and "none". The default is `action_on_pass = "none"`.

Details

This checker compares the current working directory (`getwd()`) against `required_wd`. Both paths are passed through `normalizePath()` before comparison, so differences in trailing slashes or relative vs. absolute form do not trigger a false positive. When `required_wd = NULL` (the default), there is nothing to compare against, so the check always passes; the current working directory is still reported in the message.

Value

Invisibly returns an object of class `sessioncheck_status`.

See Also

`check_attached_packages()`, `check_loaded_namespaces()`, `check_globalenv_objects()`, `check_attached_enviro`
`check_sessiontime()`, `check_required_options()`, `check_required_locale()`, `check_required_sysenv()`

Examples

```
check_working_directory(action = "message")

# a working directory that does not match the required path: reports both
# the actual and required paths
check_working_directory(action = "message", required_wd = tempdir())

# a working directory matching the required path: `action` only controls
# what happens when a problem *is* found, so use
# `action_on_pass = "message"` to confirm the clean result instead
check_working_directory(
  action = "none",
  required_wd = getwd(),
  action_on_pass = "message"
)
```

`coercion_methods` *Coerce session check object to a data frame*

Description

S3 `as.data.frame()` methods for the three classes this package defines, letting each be dropped into ordinary data frame workflows (filtering, joining, export) instead of only being inspected via `print()`/`format()`.

Usage

```
## S3 method for class 'sessioncheck_status'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'sessioncheck_sessioncheck'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'sessioncheck_sessionstate'
as.data.frame(x, row.names = NULL, optional = FALSE, which = "packages", ...)

## S3 method for class 'sessioncheck_sessionstatediff'
as.data.frame(x, row.names = NULL, optional = FALSE, which = "packages", ...)
```

Arguments

x	An object of class sessioncheck_status, sessioncheck_sessioncheck, sessioncheck_sessionstat or sessioncheck_sessionstatediff
row.names	Ignored
optional	Ignored
...	Ignored
which	For sessioncheck_sessionstate objects, which tabular component to return: one of "packages" (the default), "globalenv", or "attachments". For sessioncheck_sessionstated objects, the same three section names instead select a long-format diff table (see Details); the default is likewise "packages". Ignored for other classes.

Details

For sessioncheck_status and sessioncheck_sessioncheck objects, this coercion is lossless: every entity and status recorded in x appears as a row in the result. That guarantee does not extend to sessioncheck_sessionstate objects: sessionstate() captures more than any single rectangular table can hold, mixing scalar fields (platform, locale, matrix, document, machine, git, timing, rng), a bare character vector (libpaths), and three differently-shaped tables (packages, globalenv, attachments). as.data.frame() returns whichever one of those three tables which selects; none of the scalar fields or libpaths are represented in the result. Use x$platform, x$machine, xgit, xlibpaths, etc. (or unclass(x) for everything at once) to access those directly.

sessioncheck_sessionstatediff objects (from compare_sessionstates()) coerce differently again: each which selects a single long-format table with one row per key (package/name/name, for "packages"/"globalenv"/"attachments" respectively) and tracked field, with columns <key>, change ("added", "removed", or "modified"), field, old, and new. A key present in only one snapshot contributes one row per tracked field, all with the same change, and old/new NA on whichever side it didn't exist; a key present in both snapshots contributes a row only for fields that actually changed. The tracked fields are attached/ondisk_version/loaded_version/source for "packages", class/size/hash for "globalenv", and type for "attachments" – the same fields compare_sessionstates() tracks for its own modified tables. "globalenv" additionally has a verified column (NA for "added"/"removed" rows, since there is nothing to verify when a key only exists in one snapshot; TRUE/FALSE for "modified" rows – see compare_sessionstates()

for what verified means). "attachments" never has "modified" rows, since a type change for an existing search-path entry isn't a realistic scenario.

Value

A data frame

`compare_sessionstates` *Compare two session state snapshots*

Description

`compare_sessionstates()` reports how two `sessionstate()` snapshots differ. This is the comparison counterpart to `sessionstate()`'s point-in-time capture: take a snapshot with `sessionstate()`, do some work, take a second snapshot, and pass both to `compare_sessionstates()` to see what changed.

Usage

```
compare_sessionstates(old, new)
```

Arguments

`old`	A `sessioncheck_sessionstate` object (from `sessionstate()`), treated as the baseline.
`new`	A `sessioncheck_sessionstate` object (from `sessionstate()`), treated as the later snapshot.

Details

`sessionstate()`'s 12 elements fall into four shapes, and each is diffed differently:

- **Record** elements (`platform`, `locale`, `matrix`, `document`, `machine`, `git`, `rng`) are named lists of scalar fields. Each is diffed field-by-field via `identical()`, producing a data frame with columns `field`, `old`, `new`, and `changed`.
- `timing` is a record element, but `captured_at`/`elapsed_sec` necessarily differ between any two calls to `sessionstate()`, so flagging them as "changed" the way other fields are would be noise every time. Instead `timing` reports `captured_at_old`, `captured_at_new`, `wall_elapsed` (the difference between the two capture times, in seconds), and `uptime_elapsed` (the difference between the two `elapsed_sec` values). The two are usually equal; a mismatch (e.g. the machine slept between snapshots) is itself worth noticing.
- `libpaths` is a plain character vector, diffed via `setdiff()` in both directions: `list(added = ..., removed = ...)`. Paths present in both snapshots but reordered are not reported as a change.

- **Keyed table** elements (packages, globalenv, attachments) are data frames. Each is diffed into list(added = <data frame>, removed = <data frame>, modified = <data frame>) (attachments has no modified table – a type change for the same search-path entry isn't a realistic scenario). added/removed are rows present in only one snapshot (keyed by package/name/name respectively); modified covers rows present in both where a tracked column differs, in a long format with one row per changed field (package/name, field, old, new for packages; see below for globalenv's slightly different modified columns).

Keyed-table diffing assumes each key (package for packages; name for globalenv/attachments) appears at most once per snapshot – true for anything sessionstate() itself produces. old/new are checked for this on every keyed-table section, and compare_sessionstates() errors with an informative message identifying the offending snapshot, section, and duplicated value(s) if it doesn't hold (e.g. for a hand-constructed or corrupted sessioncheck_sessionstate object).

Keyed-table diffing is purely key-based: it has no way to detect a rename. A package or global environment object that is renamed but otherwise unchanged between old and new (e.g. pkgA reinstalled under a new name, or x renamed to y via assign()) is reported as one removed row (the old key) plus one added row (the new key), never as a single "renamed" entry – there is no general way to tell a rename apart from an unrelated removal-plus-addition that happens to involve similar values. This is inherent to any key-based diff, not a bug to be fixed.

globalenv's modified table relies on the hash column sessionstate() records for each object (an MD5 fingerprint of the object's serialized value). When both snapshots have a non-NA hash for an object, a hash mismatch is what marks it modified (verified = TRUE); when either side's hash is NA (the object couldn't be serialized – see sessionstate()'s Global environment section), the comparison falls back to class/size only, and the row is marked verified = FALSE to be explicit that a value change could have gone undetected. If an object's hash goes from NA to non-NA or vice versa between snapshots – e.g. it shrank below sessionstate_hash_max_size, or started/stopped failing to serialize – that is reported as its own "hash" row (with verified = FALSE), even when class/size are unchanged, since the object's verifiability itself changed.

verified = TRUE means the hash comparison itself is trustworthy as far as R's serialization can see – it does not mean every possible kind of change is detectable. For an object that is a thin wrapper around state living outside R's memory (e.g. a database connection, an Arrow Table/RecordBatchReader, a magick image; see sessionstate()'s Global environment section), hash fingerprints the R-level wrapper, typically a fixed placeholder for the underlying pointer, not the external data. A verified = TRUE, unchanged-hash result for such an object means "unchanged as far as R can observe", not "definitely unchanged" – the external state could have changed without the R-level object being reassigned. This is inherent to hashing via R-level serialization, not a defect in the comparison logic.

A related, opposite-direction limitation: serialize()'s traversal of an environment's bindings is order-dependent, not purely content-dependent (see sessionstate()'s Global environment section). For an object that is, or contains, an environment – an R6 object, a closure, a reference class instance – this can produce a hash mismatch, and so a false modified row here, even when the object's actual contents are unchanged. verified = TRUE does not rule this out.

A warning is issued if new$timing$captured_at is earlier than old$timing$captured_at, since that usually means the two arguments were passed in the wrong order; the comparison is still computed either way.

Value

An object of class `sessioncheck_sessionstatediff`, a list with the same 12 elements as `sessionstate()` (`platform`, `locale`, `matrix`, `document`, `machine`, `git`, `timing`, `rng`, `libpaths`, `packages`, `globalenv`, `attachments`), each holding a diff rather than a raw snapshot. See Details for the shape of each element.

See Also

`sessionstate()`

Examples

```
baseline <- sessionstate()
# assign() into .GlobalEnv explicitly (rather than `x <- 1:10`) so this
# example is correct wherever it's evaluated: sessionstate() specifically
# inspects .GlobalEnv, but some example/doc runners (e.g. pkgdown) do not
# evaluate example code there
assign("sessioncheck_example_obj", 1:10, envir = .GlobalEnv)
current <- sessionstate()
compare_sessionstates(baseline, current)
rm(sessioncheck_example_obj, envir = .GlobalEnv)
```

---

display_methods *Format and print sessioncheck objects*

Description

S3 `format()`/`print()` methods for the three classes this package defines. `sessioncheck_status`/`sessioncheck_sessionc` objects render as a one-line-per-check status summary; `sessioncheck_sessionstate` objects render as a multi-section report, and the arguments below let that report be filtered down to specific fields or columns per section.

Usage

```
## S3 method for class 'sessioncheck_status'
format(x, ...)

## S3 method for class 'sessioncheck_sessioncheck'
format(x, ...)

## S3 method for class 'sessioncheck_status'
print(x, ...)

## S3 method for class 'sessioncheck_sessioncheck'
print(x, ...)

## S3 method for class 'sessioncheck_sessionstate'
```

```
format(
  x,
  platform = NULL,
  locale = NULL,
  matrix = NULL,
  document = NULL,
  machine = NULL,
  git = NULL,
  timing = NULL,
  rng = NULL,
  packages = NULL,
  globalenv = NULL,
  globalenv_n = NULL,
  attachments = NULL,
  ...
)

## S3 method for class 'sessioncheck_sessionstate'
print(
  x,
  platform = NULL,
  locale = NULL,
  matrix = NULL,
  document = NULL,
  machine = NULL,
  git = NULL,
  timing = NULL,
  rng = NULL,
  packages = NULL,
  globalenv = NULL,
  globalenv_n = NULL,
  attachments = NULL,
  ...
)

## S3 method for class 'sessioncheck_sessionstatediff'
format(
  x,
  changed_only = NULL,
  packages = NULL,
  globalenv = NULL,
  attachments = NULL,
  max_rows = NULL,
  ...
)

## S3 method for class 'sessioncheck_sessionstatediff'
print(
```

```
  x,
  changed_only = NULL,
  packages = NULL,
  globalenv = NULL,
  attachments = NULL,
  max_rows = NULL,
  ...
)
```

Arguments

x	An object of class sessioncheck_status, sessioncheck_sessioncheck, sessioncheck_sessionstat or sessioncheck_sessionstatediff
...	Ignored
platform	For sessioncheck_sessionstate objects, an optional character vector selecting which platform fields to display (from "version", "os", "system", "ui", "tz", "date"). Defaults to showing all fields. Ignored for other classes. See Details for how the default is resolved.
locale	For sessioncheck_sessionstate objects, an optional character vector selecting which locale fields to display (from "language", "collate", "ctype"). Defaults to showing all fields. Ignored for other classes. See Details for how the default is resolved.
matrix	For sessioncheck_sessionstate objects, an optional character vector selecting which matrix-products fields to display (from "blas", "lapack"). Defaults to showing all fields. Ignored for other classes. See Details for how the default is resolved.
document	For sessioncheck_sessionstate objects, an optional character vector selecting which document-products fields to display (from "pandoc", "quarto"). Defaults to showing all fields. Ignored for other classes. See Details for how the default is resolved.
machine	For sessioncheck_sessionstate objects, an optional character vector selecting which machine fields to display (from "nodename", "user", "cwd"). Defaults to showing all fields. Ignored for other classes. See Details for how the default is resolved.
git	For sessioncheck_sessionstate objects, an optional character vector selecting which git fields to display (from "sha", "dirty"). Defaults to showing all fields. Ignored for other classes. See Details for how the default is resolved.
timing	For sessioncheck_sessionstate objects, an optional character vector selecting which timing fields to display (from "captured_at", "elapsed_sec"). Defaults to showing all fields. Ignored for other classes. See Details for how the default is resolved.
rng	For sessioncheck_sessionstate objects, an optional character vector selecting which RNG fields to display (from "kind", "normal_kind", "sample_kind", "seed_hash"). Defaults to showing all fields. Ignored for other classes. See Details for how the default is resolved.

packages — For sessioncheck_sessionstate objects, an optional character vector selecting which package inventory columns to display (see sessionstate() for the full list of columns). Defaults to c("package", "attached", "loaded_version", "source"). For sessioncheck_sessionstatediff objects, the same column selection and default apply to the added/removed blocks of the "Packages" section (the modified block always shows its own fixed field/old/new columns, so packages does not affect it). Ignored for other classes. See Details for how the default is resolved.

globalenv — For sessioncheck_sessionstate objects, an optional character vector selecting which global environment columns to display (from "name", "class", "size", "hash"). Defaults to c("name", "class", "size") (omitting "hash", a long fingerprint mainly useful programmatically – see compare_sessionstates()). For sessioncheck_sessionstatediff objects, the same column selection and default apply to the added/removed blocks of the "Global environment" section (the modified block is unaffected – see packages above). Ignored for other classes. See Details for how the default is resolved, and for how globalenv_n separately controls the number of rows shown for sessioncheck_sessionstate objects.

globalenv_n — For sessioncheck_sessionstate objects, an optional single number giving the maximum number of globalenv rows to display, largest objects first. Defaults to 10. Ignored for other classes. See Details for how the default is resolved.

attachments — For sessioncheck_sessionstate objects, an optional character vector selecting which attached-environment columns to display (from "name", "type"). Defaults to showing all columns. For sessioncheck_sessionstatediff objects, the same column selection applies to the added/removed blocks of the "Attached environments" section (which has no modified block at all – see compare_sessionstates()). Ignored for other classes. See Details for how the default is resolved.

changed_only — For sessioncheck_sessionstatediff objects, whether to collapse sections/fields with no detected change down to a single "(no changes)" line (TRUE by default) or always show every field. Ignored for other classes. See Details for how the default is resolved.

max_rows — For sessioncheck_sessionstatediff objects, an optional single number giving the maximum number of rows to display in each added/removed/modified block of the "Packages", "Global environment", and "Attached environments" sections. Defaults to 10. Ignored for other classes. See Details for how the default is resolved.

Details

For sessioncheck_sessionstate objects, the platform/locale/matrix/ document/machine/git/timing/rng/packages/g globalenv_n/attachments arguments are resolved through the same precedence used elsewhere in the package: an explicit argument always wins; otherwise, getOption("sessioncheck") is checked for a sessionstate_platform, sessionstate_locale, sessionstate_matrix, sessionstate_document, sessionstate_machine, sessionstate_git, sessionstate_timing, sessionstate_rng, sessionstate_packages, sessionstate_globalenv, sessionstate_globalenv_n, or sessionstate_attachments field

(respectively); if neither is set, a built-in default is used (showing every field/column, except for packages, which defaults to c("package", "attached", "loaded_version", "source"), globalenv, which defaults to c("name", "class", "size"), and globalenv_n, which defaults to 10). This selection only affects what is displayed; it never changes the underlying object, so as.data.frame() always returns the full package inventory, and x$globalenv/x$attachments always return their full data frames, regardless of any selection in effect.

For sessioncheck_sessionstatediff objects, changed_only/packages/ globalenv/attachments/max_rows are resolved through the same precedence: an explicit argument always wins; otherwise getOption("sessioncheck") is checked for a sessionstatediff_changed_only, sessionstatediff_packages, sessionstatediff_globalenv, sessionstatediff_attachments, or sessionstatediff_max_rows field (respectively); if neither is set, a built-in default is used: TRUE for changed_only, 10 for max_rows, and the same packages/globalenv/attachments defaults as sessioncheck_sessionstate objects use (see above). max_rows applies independently to every added/removed/modified block, and does not affect the underlying object – as.data.frame() on a sessioncheck_sessionstatediff always returns every row.

Value

Character vector

`sessioncheck` *Checks the overall status of the R session*

Description

sessioncheck() is the top-level orchestrator for this package's individual session checks: it runs one or more check_*() functions (e.g. check_attached_packages(), check_globalenv_objects()) in a single call and combines their results. Like the individual checks it wraps, it can produce errors, warnings, or messages if requested.

Usage

```
sessioncheck(action = NULL, checks = NULL, action_on_pass = NULL, ...)
```

Arguments

action	Behavior to take if the status is not clean. Possible values are "error", "warn", "message", and "none". If the user does not specify an action, the default is action = "warn".
checks	Character vector listing the checks to run. If the user does not specify the checks, the default is to run checks = c("globalenv_objects", "attached_packages", "attached_environments").
action_on_pass	Behavior to take if the status is clean. Possible values are "message" and "none". If the user does not specify a value, the default is action_on_pass = "none".
...	Arguments passed to individual checks.

Details

The following arguments are recognized via ...:

- allow_globalenv_objects is passed to check_globalenv_objects()
- allow_attached_packages is passed to check_attached_packages()
- allow_attached_environments is passed to check_attached_environments()
- allow_loaded_namespaces is passed to check_loaded_namespaces()
- max_sessiontime is passed to check_sessiontime()
- required_options is passed to check_required_options()
- required_locale is passed to check_required_locale()
- required_sysenv is passed to check_required_sysenv()
- required_wd is passed to check_working_directory()

Other arguments are ignored.

Value

Invisibly returns an object of class sessioncheck_sessioncheck.

See Also

check_attached_packages(), check_loaded_namespaces(), check_globalenv_objects(), check_attached_enviro
check_sessiontime(), check_required_options(), check_required_locale(), check_required_sysenv(),
check_working_directory()

Examples

```
sessioncheck(action = "message")

# a session with nothing flagged by the default checks: `action` only
# controls what happens when a problem *is* found, so pass
# `action_on_pass = "message"` to confirm the clean result instead
sessioncheck(
  action = "none",
  allow_globalenv_objects = ls(envir = .GlobalEnv, all.names = TRUE),
  allow_attached_packages = .packages(),
  allow_attached_environments = search(),
  action_on_pass = "message"
)
```

`sessionstate` *Report the current state of the R session*

Description

`sessionstate()` captures a point-in-time, human-readable snapshot of the R session: platform details, selected machine information, session timing, an inventory of attached and loaded-namespace packages (including remote source tracking for packages installed from GitHub), the contents of the global environment, and the non-package entries on the search path. It is intended as a companion to `sessioncheck()`: where `sessioncheck()` is typically called at the *start* of a script to check for a clean session, `sessionstate()` is intended to be called at the *end* of a script to produce an audit log of the environment the script actually ran in.

Usage

```
sessionstate()
```

Value

An object of class `sessioncheck_sessionstate`, a list with elements `platform`, `locale`, `matrix`, `document`, `machine`, `git`, `timing`, `rng`, `libpaths`, `packages`, `globalenv`, and `attachments`.

Platform

The `platform` element records `version` (the running R version, via `R.version.string`), `os` (the operating system, preferring `utils::osVersion()` when available and falling back to `Sys.info()`), `system` (the R build's `R.version$system`), `ui` (the interface running the session – `"non-interactive"` when `interactive()` is `FALSE`, otherwise the frontend reported by `.Platform$GUI`, e.g. `"RStudio"` or `"Positron"`), `tz` (the session timezone via `Sys.timezone()`), and `date` (the capture date).

Locale

The `locale` element records `language`, `collate`, and `ctype`. These are split out from `platform` because they describe how text and dates are formatted for this session, rather than what/where/when the session is running.

Matrix

The `matrix` element records `blas` and `lapack`, the shared libraries backing R's linear algebra routines (as reported by `extSoftVersion()` and `La_library()`). Like `locale`, this is split out from `platform` – in this case mirroring how base R's `utils::sessionInfo()` treats "Matrix products" as its own block rather than nesting it under platform info.

Document

The document element's pandoc and quarto fields record the versions of those two document-rendering tools, if found (NA otherwise). Both checks prefer the IDE-provided location over whatever happens to be on PATH (RSTUDIO_PANDOC for pandoc, QUARTO_PATH for quarto), since RStudio/Positron bundle their own copies that may differ from a separately installed one. Deliberately not tracked: other system dependencies (e.g. LaTeX, Hugo, spatial libraries) are package-specific rather than session-wide, and tracking them well would mean tracking many of them; pandoc/quarto are included because they, like BLAS/LAPACK, are already tracked by `utils::sessionInfo()` or `sessioninfo::session_info()`. document has no sessionInfo() precedent (unlike matrix), but is grouped the same way for consistency.

Machine

The machine element includes the node name and user reported by `Sys.info()`, along with the working directory reported by `getwd()` at capture time (cwd) – useful for a reproducibility audit since relative paths used elsewhere in a script only resolve correctly relative to this directory. Because this can reveal a hostname, local username, or directory structure, be mindful about where sessionstate() output is stored or shared. The same caution applies to the ondisk_path/loaded_path columns of packages, since library paths often embed a home directory.

Git

The git element records sha, the current commit (`git rev-parse HEAD`, run in the working directory captured as machine$cwd), and dirty, whether the working tree has uncommitted changes (`git status --porcelain` is non-empty). Both are NA if the working directory isn't inside a git repository, or if git itself isn't installed. This is arguably the single most useful field for reproducing a script's output later: sha identifies exactly which version of the code ran, and dirty flags whether that identification is trustworthy (a TRUE means the code that ran may not match any commit).

Timing

The timing element records captured_at, the capture time reported by `Sys.time()`, and elapsed_sec, the session's elapsed run time in seconds (the "elapsed" component of `proc.time()`). Together they let an audit log show both when a snapshot was taken and how long the session had already been running at that point.

RNG

The rng element records `RNGkind()` (as kind, normal_kind, and sample_kind) together with seed_hash, an MD5 fingerprint of .Random.seed (via `tools::md5sum()`, since base R has no in-memory hashing function). seed_hash is NA if the RNG hasn't been used yet this session (nothing has consumed a random draw, so .Random.seed doesn't exist); sessionstate() never forces this into existence, since doing so would itself consume a draw as a side effect of an audit call. The hash exists to make RNG state comparable across renders without printing the seed itself: for example, comparing seed_hash between two rendered versions of the same Quarto/R Markdown document shows whether an edit changed the RNG state anywhere upstream, without having to inspect or store the (long, not directly meaningful) seed value.

Library paths

The libpaths element is the character vector returned by .libPaths(), i.e. the library locations R searches, in search order. It complements packages: that element records where each individual package resolved *to* (ondisk_path), while libpaths records where R was looking in the first place, which matters when, e.g., a project-local library shadows a personal one. Unlike the other elements, there is no corresponding display-filtering argument for libpaths, since it is already a flat list of paths rather than a set of named fields or columns to choose among; it is always shown in full.

Packages

The packages element covers every package that is either attached to the search path or loaded via namespace (i.e., union(.packages(), loadedNamespaces())). It has columns package, attached, ondisk_version (the version recorded in the installed package's DESCRIPTION file), loaded_version (the version of the namespace actually loaded into memory), version_mismatch (TRUE when the two disagree, e.g. because the package was updated on disk after this session loaded it), ondisk_path and loaded_path (the library paths a package currently resolves to versus where its loaded namespace actually came from), path_mismatch (TRUE when both exist but disagree, e.g. after a .libPaths() change mid-session), removed_from_disk (TRUE when the namespace is loaded but no longer found on disk at all), and source, which classifies each package as "base", "CRAN (R x.y.z)", "Github (user/repo@sha)", another remote type, or "local" when no remote metadata is available.

Global environment

The globalenv element is a data frame with one row per object in .GlobalEnv (including dot-prefixed objects), with columns name, class, size (in bytes, as reported by utils::object.size()), and hash (an MD5 fingerprint of the object's serialized value, in the same spirit as rng$seed_hash: serialize() the object, then run tools::md5sum() on the result). hash is NA when an object cannot be serialized at all, which is rare in practice – objects backed by an external pointer (e.g. a database connection) typically still serialize to a placeholder rather than erroring. This is reported rather than silently treated as "unchanged" by anything comparing two snapshots. Only object names, classes, sizes, and value fingerprints are captured, never values themselves. Because a long-running script can accumulate many objects, the default display shows only the largest few, and omits hash (see "Selecting which elements are displayed" below); the captured object itself always holds every object and every column.

Hashing cost scales linearly with an object's size (roughly 5-6 seconds per GB, dominated by serialize() itself rather than the disk I/O tools::md5sum() requires). For a workspace holding very large objects (e.g. multi-GB models or data frames), this can add a noticeable amount of time to a single sessionstate() call. Setting a sessionstate_hash_max_size field (a number of bytes) via options(sessioncheck = list(...)) caps this: any object larger than the limit gets hash = NA instead of being serialized at all, using the same "not verifiable" semantics as an object that fails to serialize (see above). There is no corresponding function argument – sessionstate() takes none – so this is resolved purely as option-or-default (Inf by default, i.e. no size limit and no change to prior behavior).

A related but distinct limitation: some R objects are thin wrappers around state that lives outside R's memory entirely – a magick image, an Arrow Table or RecordBatchReader, a database connection, a memory-mapped file. Hashing such an object only fingerprints its R-level representation –

typically a fixed placeholder for the underlying pointer itself, per `serialize()`'s handling of external pointers (see `compare_sessionstates()`'s Details) – not the external data it points to. If that external state changes without the R-level object itself being reassigned (e.g. writing to a database connection, advancing a stream's read position, mutating a file the object references), `hash` can stay unchanged even though the object's real, externally-held content did not. This is a limitation of what `sessionstate()` can observe from within R, not a defect in the hashing itself: it never inspects state outside R's memory, and so cannot distinguish "genuinely unchanged" from "changed only outside R" for objects like these.

A different limitation runs in the opposite direction: for an object that is, or contains, an environment – an R6 object, a closure (via its enclosing environment), a reference class instance – `serialize()`'s traversal of an environment's bindings depends on insertion history, not only on the environment's current contents. Two environments holding identical bindings, populated in a different order, can therefore serialize (and hash) differently even though nothing about them has meaningfully changed. Where the external-pointer limitation above can hide a real change (a false negative), this one can report a change that never happened (a false positive) in `compare_sessionstates()`'s `globalenv$modified` table. There is no general fix for this within base R's `serialize()`; avoiding it would require a custom, order-independent serialization of environment-backed objects, which `sessionstate()` does not attempt.

Attachments

The `attachments` element is a data frame with one row per entry on the search path (as returned by `search()`), with columns `name` and `type` (`"package"` or `"other"`). This surfaces non-package attachments (e.g. `tools:rstudio`, or environments added via `attach()`) that aren't reflected in `packages`.

Selecting which elements are displayed

`sessionstate()` itself always captures every field in full (`globalenv` is never truncated at capture time). To display only a subset when printing, pass `platform`/`locale`/`matrix`/`document`/`machine`/`git`/`timing`/`rng`/`packages`/`globalenv`/`attachments` arguments to `print()` or `format()` on the result, or set defaults via `options(sessioncheck = list(sessionstate_packages = ...))` (see display_methods for the full precedence rules and option names). The `globalenv_n` argument separately controls how many rows of `globalenv` are shown (largest objects first), independent of which columns are selected. None of this affects the underlying object, so `x$globalenv`/`x$attachments` always return their full data frames. Separately, `as.data.frame()` returns one of the three tables captured by `sessionstate()` (`packages`, `globalenv`, or `attachments`, selected via its `which` argument); see coercion_methods for why this coercion, unlike the one for `sessioncheck()`, cannot be lossless.

See Also

`sessioncheck()`

Examples

```
sessionstate()
```

Index