Package: multiverse.internals (via r-universe)

June 23, 2024

Title Internal Infrastructure for R-multiverse

Description R-multiverse requires this internal internal
infrastructure package to automate contribution reviews and
populate universes.

Version 0.2.4

License MIT + file LICENSE

URL `https://github.com/r-multiverse/multiverse.internals`

BugReports `https://github.com/r-multiverse/multiverse.internals/issues`

Depends R (>= 3.5.0)

Imports gh, igraph, jsonlite, nanonext, pkgsearch, utils, vctrs

Suggests testthat (>= 3.0.0)

Encoding UTF-8

Language en-US

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Repository https://r-multiverse.r-universe.dev

RemoteUrl https://github.com/r-multiverse/multiverse.internals

RemoteRef 0.2.4

RemoteSha b09acbbcff18c157b13853ea24751b48c70c8f9c

Contents

`issues_checks` *Report issues from R-universe package check results.*

Description

Check R-universe package check results.

Usage

```
issues_checks(meta = meta_checks())
```

Arguments

`meta`	A data frame with R-universe package check results returned by `meta_checks()`.

Details

`issues_checks()` reads output from the R-universe check API to scan all R-multiverse packages for issues that may have happened during building and testing.

Value

A named list of information about packages which do not comply with `DESCRPTION` checks. Each name is a package name, and each element contains specific information about non-compliance.

Package issues

Functions like `issues_versions()` and `issues_descriptions()` perform health checks for all packages in R-multiverse. Only packages that pass these checks go to the production repository at `https://production.r-multiverse.org`. For a complete list of checks, see the `issues_*()` functions listed at `https://r-multiverse.org/multiverse.internals/reference.html`. `record_versions()` updates the version number history of releases in R-multiverse, and `record_issues()` gathers together all the issues about R-multiverse packages.

See Also

Other issues: `issues_dependencies()`, `issues_descriptions()`, `issues_versions()`

Examples

```
meta <- meta_checks(repo = "https://wlandau.r-universe.dev")
issues <- issues_checks(meta = meta)
str(issues)
```

`issues_dependencies` *Report package dependency issues*

Description

Flag packages which have issues in their strong dependencies (`Imports:`, `Depends:`, and `LinkingTo:` in the `DESCRIPTION`.) These include indirect/upstream dependencies, as well, not just the explicit mentions in the `DESCRIPTION` file.

Usage

```
issues_dependencies(packages, meta = meta_packages(), verbose = FALSE)
```

Arguments

`packages`	Character vector of names of packages with other issues.
`meta`	A data frame with R-universe package check results returned by `meta_checks()`.
`verbose`	`TRUE` to print progress while checking issues with dependencies, `FALSE` otherwise.

Value

A nested list of problems triggered by dependencies. The names of top-level elements are packages affected downstream. The value of each top-level element is a list whose names are Each element of this inner list is a character vector of relevant dependencies of the downstream package.

For example, consider a linear dependency graph where `crew.cluster` depends on `crew`, `crew` depends on `mirai`, and `mirai` depends on `nanonext`. We represent the graph like this: `nanonext -> mirai -> crew -> crew.cluster`. If `nanonext` has an issue, then `issues_dependencies()` returns `list(crew.cluster = list(nanonext = "crew"), ...)`, where `...` stands for additional named list entries. From this list, we deduce that `nanonext` is causing an issue affecting `crew.cluster` through the direct dependency on `crew`.

The choice in output format from `issues_dependencies()` allows package maintainers to more easily figure out which direct dependencies are contributing issues and drop those direct dependencies if necessary.

Package issues

Functions like `issues_versions()` and `issues_descriptions()` perform health checks for all packages in R-multiverse. Only packages that pass these checks go to the production repository at `https://production.r-multiverse.org`. For a complete list of checks, see the `issues_*()` functions listed at `https://r-multiverse.org/multiverse.internals/reference.html`. `record_versions()` updates the version number history of releases in R-multiverse, and `record_issues()` gathers together all the issues about R-multiverse packages.

See Also

Other issues: `issues_checks()`, `issues_descriptions()`, `issues_versions()`

Examples

```
meta <- meta_packages(repo = "https://wlandau.r-universe.dev")
issues_dependencies(packages = character(0L), meta = meta)
issues_dependencies(packages = "crew.aws.batch", meta = meta)
issues_dependencies(packages = "nanonext", meta = meta)
issues_dependencies(packages = "crew", meta = meta)
issues_dependencies(packages = c("crew", "mirai"), meta = meta)
```

issues_descriptions *Report* DESCRIPTION *file issues.*

Description

Report issues with the DESCRIPTION files of packages.

Usage

```
issues_descriptions(meta = meta_packages())
```

Arguments

meta	A data frame with R-universe package check results returned by meta_checks().

Details

issues_descriptions() scans downloaded metadata from the PACKAGES.json file of an R universe and reports issues with a package's description file, such as the presence of a "Remotes" field.

Value

A named list of information about packages which do not comply with DESCRPTION checks. Each name is a package name, and each element contains specific information about non-compliance.

Package issues

Functions like issues_versions() and issues_descriptions() perform health checks for all packages in R-multiverse. Only packages that pass these checks go to the production repository at https://production.r-multiverse.org. For a complete list of checks, see the issues_*() functions listed at https://r-multiverse.org/multiverse.internals/reference.html. record_versions() updates the version number history of releases in R-multiverse, and record_issues() gathers together all the issues about R-multiverse packages.

See Also

Other issues: issues_checks(), issues_dependencies(), issues_versions()

Examples

```
meta <- meta_packages(repo = "https://wlandau.r-universe.dev")
issues <- issues_descriptions(meta = meta)
str(issues)
```

`issues_versions` *Check package versions.*

Description

Check package version number history for compliance.

Usage

```
issues_versions(versions)
```

Arguments

`versions`	Character of length 1, file path to a JSON manifest tracking the history of released versions of packages. The official versions file for R-multiverse is maintained and updated periodically at `https://github.com/r-multiverse/checks/blob/main/versions.json`.

Details

This function checks the version number history of packages in R-multiverse and reports any packages with issues. The current released version of a given package must be unique, and it must be greater than all the versions of all the previous package releases.

Value

A named list of information about packages which do not comply with version number history checks. Each name is a package name, and each element contains specific information about version non-compliance: the current version number, the current version hash, and the analogous versions and hashes of the highest-versioned release recorded.

Package issues

Functions like `issues_versions()` and `issues_descriptions()` perform health checks for all packages in R-multiverse. Only packages that pass these checks go to the production repository at `https://production.r-multiverse.org`. For a complete list of checks, see the `issues_*()` functions listed at `https://r-multiverse.org/multiverse.internals/reference.html`. `record_versions()` updates the version number history of releases in R-multiverse, and `record_issues()` gathers together all the issues about R-multiverse packages.

See Also

Other issues: `issues_checks()`, `issues_dependencies()`, `issues_descriptions()`

Examples

```
# See https://github.com/r-multiverse/checks/blob/main/versions.json
# for the official versions JSON for R-multiverse.
lines <- c(
  "[",
  " {",
  " \"package\": \"package_unmodified\",",
  " \"version_current\": \"1.0.0\",",
  " \"hash_current\": \"hash_1.0.0\",",
  " \"version_highest\": \"1.0.0\",",
  " \"hash_highest\": \"hash_1.0.0\"",
  " },",
  " {",
  " \"package\": \"version_decremented\",",
  " \"version_current\": \"0.0.1\",",
  " \"hash_current\": \"hash_0.0.1\",",
  " \"version_highest\": \"1.0.0\",",
  " \"hash_highest\": \"hash_1.0.0\"",
  " },",
  " {",
  " \"package\": \"version_incremented\",",
  " \"version_current\": \"2.0.0\",",
  " \"hash_current\": \"hash_2.0.0\",",
  " \"version_highest\": \"2.0.0\",",
  " \"hash_highest\": \"hash_2.0.0\"",
  " },",
  " {",
  " \"package\": \"version_unmodified\",",
  " \"version_current\": \"1.0.0\",",
  " \"hash_current\": \"hash_1.0.0-modified\",",
  " \"version_highest\": \"1.0.0\",",
  " \"hash_highest\": \"hash_1.0.0\"",
  " }",
  "]"
)
versions <- tempfile()
writeLines(lines, versions)
out <- issues_versions(versions)
str(out)
```

---

meta_checks *List metadata about R-universe package checks*

Description

List package checks results reported by the R-universe package API.

Usage

```
meta_checks(repo = "https://community.r-multiverse.org")
```

Arguments

`repo`	Character of length 1, URL of the package repository. R-multiverse uses `"https://community.r-multi`

Value

A data frame with one row per package and columns with package check results.

Examples

```
meta_checks(repo = "https://wlandau.r-universe.dev")
```

`meta_packages` *List package metadata*

Description

List package metadata in an R universe.

Usage

```
meta_packages(repo = "https://community.r-multiverse.org")
```

Arguments

`repo`	Character of length 1, URL of the package repository. R-multiverse uses `"https://community.r-multi`

Value

A data frame with one row per package and columns with package metadata.

Examples

```
meta_packages(repo = "https://wlandau.r-universe.dev")
```

`record_issues` *Record package issues.*

Description

Record R-multiverse package issues in package-specific JSON files.

Usage

```
record_issues(
  repo = "https://community.r-multiverse.org",
  versions = "versions.json",
  output = "issues",
  mock = NULL,
  verbose = FALSE
)
```

Arguments

`repo`	Character of length 1, URL of the package repository. R-multiverse uses `"https://community.r-multi`
`versions`	Character of length 1, file path to a JSON manifest tracking the history of released versions of packages. The official versions file for R-multiverse is maintained and updated periodically at `https://github.com/r-multiverse/checks/blob/main/versions.json`.
`output`	Character of length 1, file path to the folder to record new package issues. Each call to `record_issues()` overwrites the contents of the repo.
`mock`	For testing purposes only, a named list of data frames for inputs to various intermediate functions.
`verbose`	`TRUE` to print progress while checking issues with dependencies, `FALSE` otherwise.

Value

`NULL` (invisibly).

Package issues

Functions like `issues_versions()` and `issues_descriptions()` perform health checks for all packages in R-multiverse. Only packages that pass these checks go to the production repository at `https://production.r-multiverse.org`. For a complete list of checks, see the `issues_*()` functions listed at `https://r-multiverse.org/multiverse.internals/reference.html`. `record_versions()` updates the version number history of releases in R-multiverse, and `record_issues()` gathers together all the issues about R-multiverse packages.

Issue files

For each package with observed problems, `record_issues()` writes an issue file. This issue file is a JSON list with one element per type of failing check. Each element has an informative name (for example, `checks`, `descriptions`, or `versions`) and a list of diagnostic information.

Each issue file also has a `date` field. This date the day that an issue was first noticed. It automatically resets the next time all package are resolved.

Examples

```
repo <- "https://wlandau.r-universe.dev"
output <- tempfile()
versions <- tempfile()
record_versions(
  versions = versions,
  repo = repo
)
record_issues(
  repo = repo,
  versions = versions,
  output = output
)
files <- list.files(output)
print(files)
package <- head(files, n = 1)
if (length(package)) {
  print(package)
}
if (length(package)) {
  print(readLines(file.path(output, package)))
}
```

`record_versions` *Record the manifest of package versions.*

Description

Record the manifest of versions of packages and their hashes.

Usage

```
record_versions(
  versions = "versions.json",
  repo = "https://community.r-multiverse.org",
  current = multiverse.internals::get_current_versions(repo = repo)
)
```

Arguments

`versions`	Character of length 1, file path to a JSON manifest tracking the history of released versions of packages. The official versions file for R-multiverse is maintained and updated periodically at `https://github.com/r-multiverse/checks/blob/main/versions.json`.
`repo`	Character of length 1, URL of the package repository. R-multiverse uses `"https://community.r-multi`
`current`	A data frame of current versions and hashes of packages in `repo`. This argument is exposed for testing only.

Details

This function tracks a manifest containing the current version, the current hash, the highest version ever released, and the hash of the highest version ever released. `issues_versions()` uses this information to determine whether the package complies with best practices for version numbers.

Value

`NULL` (invisibly). Writes a package version manifest and a manifest of version issues as JSON files.

Package issues

Functions like `issues_versions()` and `issues_descriptions()` perform health checks for all packages in R-multiverse. Only packages that pass these checks go to the production repository at `https://production.r-multiverse.org`. For a complete list of checks, see the `issues_*()` functions listed at `https://r-multiverse.org/multiverse.internals/reference.html`. `record_versions()` updates the version number history of releases in R-multiverse, and `record_issues()` gathers together all the issues about R-multiverse packages.

Examples

```
# R-multiverse uses https://community.r-multiverse.org as the repo.
repo <- "https://wlandau.r-universe.dev" # just for testing and examples
output <- tempfile()
versions <- tempfile()
# First snapshot:
record_versions(
  versions = versions,
  repo = repo
)
readLines(versions)
# In subsequent snapshots, we have historical information about versions.
record_versions(
  versions = versions,
  repo = repo
)
readLines(versions)
```

`review_pull_request` *Review a pull request.*

Description

Review a pull request to add packages to `packages.json`.

Usage

```
review_pull_request(owner = "r-multiverse", repo = "contributions", number)
```

Arguments

`owner`	Character of length 1, name of the package repository owner.
`repo`	Character of length 1, URL of the package repository. R-multiverse uses `"https://community.r-multiv`
`number`	Positive integer of length 1, index of the pull request in the repo.

Value

`NULL` (invisibly).

Testing

Testing of this function unfortunately needs to be manual. Test cases:

1. Add a package correctly (automatically merge).
2. Add a bad URL (manual review).
3. Change a URL (manual review).
4. Add a file in a forbidden place (close).
5. Add a custom JSON file which can be parsed (manual review).

See Also

Other pull request reviews: `review_pull_requests()`

review_pull_requests *Review pull requests.*

Description

Review pull requests which add packages to packages.json.

Usage

```
review_pull_requests(owner = "r-multiverse", repo = "contributions")
```

Arguments

owner	Character of length 1, name of the package repository owner.
repo	Character of length 1, URL of the package repository. R-multiverse uses "https://community.r-multi

Value

NULL (invisibly).

Testing

Testing of this function unfortunately needs to be manual. Test cases:

1. Add a package correctly (automatically merge).
2. Add a bad URL (manual review).
3. Change a URL (manual review).
4. Add a file in a forbidden place (close).
5. Add a custom JSON file which can be parsed (manual review).

See Also

Other pull request reviews: review_pull_request()

Index