Package: revdepcheck (via r-universe)

January 21, 2025

Title Automated Reverse Dependency Checking

Version 1.0.0.9002

Description Automated, isolated reserve dependency checking, with automatic comparison of the results to the current CRAN checks.

License MIT + file LICENSE

URL https://revdepcheck.r-lib.org, https://github.com/r-lib/revdepcheck#readme

BugReports https://github.com/r-lib/revdepcheck/issues

Imports assertthat, brio, callr, cli (>= 3.1.0), clisymbols, crancache (>= 0.0.0.9001), crayon (>= 1.4.1), curl, DBI, desc (>= 1.3.0), glue, gmailr, hms, httr, jsonlite, knitr, pkgbuild, prettyunits, processx (>= 3.3.0), progress, rcmdcheck (>= 1.3.3), rematch2, remotes (>= 2.2.0), rlang (>= 0.3.0), RSQLite, sessioninfo, tibble, utils, whoami, withr, yaml

Suggests covr, debugme, forcats, ggplot2, rmarkdown, testthat

VignetteBuilder knitr

Remotes r-lib/crancache, r-lib/remotes

Config/Needs/website tidyverse/tidytemplate

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Config/pak/sysreqs git make libssl-dev

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

RemoteUrl https://github.com/r-lib/revdepcheck

RemoteRef HEAD

RemoteSha 5a0c13819b83567a4c789311131d69dfa35a722c

Contents

`cloud_broken` *Retrieve the names broken or failed packages*

Description

Broken packages are those whose checks got worse with the dev version. Failed packages are those whose cloud jobs failed, either because the spot instance was shut down by AWS or because the checks used too much memory and were killed.

Usage

```
cloud_broken(
  job_name = cloud_job(pkg = pkg),
  pkg = ".",
  install_failures = FALSE,
  timeout_failures = FALSE
)

cloud_failed(job_name = cloud_job(pkg = pkg), pkg = ".")
```

Arguments

job_name	The job name, as returned by cloud_check().
pkg	Path to package.
install_failures	Whether to include packages that failed to install.
timeout_failures	Whether to include packages that timed out.

Value

A character vector with the names of broken packages, to be passed to cloud_check().

See Also

Other cloud: cloud_browse(), cloud_cancel(), cloud_check(), cloud_details(), cloud_fetch_results(), cloud_plot(), cloud_report(), cloud_results(), cloud_status(), cloud_summary()

---

cloud_browse *Browse to the AWS url for the job*

Description

This is useful for closer inspection of individual jobs while they are running or after the fact.

Usage

```
cloud_browse(job_name = cloud_job(), package = NULL)
```

Arguments

job_name	The job name, as returned by cloud_check().
package	If NULL browses to the URL of the overall job. If a package name, browses to the URL for that specific package job.

See Also

Other cloud: cloud_broken(), cloud_cancel(), cloud_check(), cloud_details(), cloud_fetch_results(), cloud_plot(), cloud_report(), cloud_results(), cloud_status(), cloud_summary()

cloud_cancel *Cancel a running cloud run*

Description

Cancel a running cloud run

Usage

```
cloud_cancel(job_name = cloud_job())
```

Arguments

job_name	The job name, as returned by cloud_check().

See Also

Other cloud: cloud_broken(), cloud_browse(), cloud_check(), cloud_details(), cloud_fetch_results(), cloud_plot(), cloud_report(), cloud_results(), cloud_status(), cloud_summary()

cloud_check *Submit a reverse dependency checking job to the cloud*

Description

Submit a reverse dependency checking job to the cloud

Usage

```
cloud_check(
  pkg = ".",
  tarball = NULL,
  revdep_packages = NULL,
  extra_revdeps = NULL,
  r_version = "4.3.1",
  check_args = "--no-manual",
  bioc = FALSE
)
```

Arguments

pkg	Path to package.
tarball	A pre-built package tarball, if NULL a tarball will be automatically built for the package at pkg by pkgbuild::build().
revdep_packages	A character vector of packages to check, if NULL equal to cran_revdeps()
extra_revdeps	Additional packages to use as source for reverse dependencies.
r_version	The R version to use.
check_args	Additional argument to pass to R CMD check
bioc	Also check revdeps that live in Bioconductor? Default FALSE. Note that the cloud revdep check service does not currently include system dependencies of Bioconductor packages, so there is potential for more failed checks.

Value

The AWS Batch job name

See Also

Other cloud: cloud_broken(), cloud_browse(), cloud_cancel(), cloud_details(), cloud_fetch_results(), cloud_plot(), cloud_report(), cloud_results(), cloud_status(), cloud_summary()

---

cloud_details *Display detailed revdep results from a cloud run*

Description

Display detailed revdep results from a cloud run

Usage

```
cloud_details(job_name = cloud_job(pkg = pkg), revdep, pkg = ".")
```

Arguments

job_name	The job name, as returned by cloud_check().
revdep	Name of the revdep package
pkg	Path to package.

See Also

Other cloud: cloud_broken(), cloud_browse(), cloud_cancel(), cloud_check(), cloud_fetch_results(), cloud_plot(), cloud_report(), cloud_results(), cloud_status(), cloud_summary()

`cloud_email` *Notify revdep maintainers about problems*

Description

This function uses gmail to automatically notify all maintainers of revdeps that have failures with the new version of the package. The form of the email is fixed, but it uses template parameters so that you can control the details: set the variables in `revdeps/email.yaml`. You'll be prompted to review the template before any emails are sent; or you can use `revdep_email_draft()` to see a draft version.

Usage

```
cloud_email(
  type = c("broken", "failed"),
  job_name = cloud_job(pkg = pkg),
  pkg = ".",
  packages = NULL,
  draft = FALSE
)
```

Arguments

`type`	Type of problems to notify about; either "broken" (i.e. there is a new `R CMD check` failure that did not currently occur) or "failed" (i.e. the check failure either during installation or because of a timeout).
`job_name`	The job name, as returned by `cloud_check()`.
`pkg`	Path to package.
`packages`	A character vector of package names. Use this if some emails failed to send in the previous round. If omitted uses all packages.
`draft`	If `TRUE`, create a gmail draft rather than sending the email directly.

Details

To use this function, you'll need to give the gmailr app authority to send emails from gmail. To revoke that authority, delete the `.httr-oauth` file created in your working directory.

cloud_job *Return the current cloud job*

Description

The job_name is automatically set by cloud_check() and is remembered for the duration of the current R session. If there is no active job_name, but there are local cloud check results, job_name is inferred from the most recently modified cloud check results.

Usage

```
cloud_job(job_name = NULL, pkg = ".")
```

Arguments

job_name	If not NULL, sets the active job_name to the input.
pkg	Path to package.

cloud_job_mapping *Get a tibble of batch sub-job ids for all checked packages*

Description

Get a tibble of batch sub-job ids for all checked packages

Usage

```
cloud_job_mapping(job_name = cloud_job())
```

Arguments

job_name	The job name, as returned by cloud_check().

cloud_plot *Plot the running time per package of a cloud job*

Description

Plot the running time per package of a cloud job

Usage

```
cloud_plot(job_name = cloud_job())
```

Arguments

job_name	The job name, as returned by cloud_check().

See Also

Other cloud: cloud_broken(), cloud_browse(), cloud_cancel(), cloud_check(), cloud_details(), cloud_fetch_results(), cloud_report(), cloud_results(), cloud_status(), cloud_summary()

cloud_report *Markdown report of reverse dependency check results from the cloud*

Description

You can use these functions to get intermediate reports of a running cloud check.

Usage

```
cloud_report(
  job_name = cloud_job(pkg = pkg),
  pkg = ".",
  file = "",
  all = FALSE,
  results = NULL,
  failures = TRUE
)

cloud_report_summary(
  job_name = cloud_job(pkg = pkg),
  file = "",
  all = FALSE,
  pkg = ".",
  results = NULL
)
```

```
cloud_report_problems(
  job_name = cloud_job(pkg = pkg),
  pkg = ".",
  file = "",
  all = FALSE,
  results = NULL
)

cloud_report_failures(
  job_name = cloud_job(pkg = pkg),
  pkg = ".",
  file = "",
  results = NULL
)

cloud_report_cran(job_name = cloud_job(pkg = pkg), pkg = ".", results = NULL)
```

Arguments

job_name	The job name, as returned by cloud_check().
pkg	Path to package.
file	File to write output to. Default will write to console.
all	Whether to report all problems, including the ones that were already present in the old version of the package. This potentially generated a lot of output, most of which was irrelevant, so they are omitted by default, and only problems seen with the new version of the package are reported.
results	Results from cloud_results(). Expert use only.
failures	Save failures to disk?

See Also

Other cloud: cloud_broken(), cloud_browse(), cloud_cancel(), cloud_check(), cloud_details(), cloud_fetch_results(), cloud_plot(), cloud_results(), cloud_status(), cloud_summary()

---

cloud_status *Monitor the status of a cloud job*

Description

The format of the status bar is [jobs_queued/jobs_running/jobs_succeeded/jobs_failed - total_jobs] time_elaps

Usage

```
cloud_status(job_name = cloud_job(), update_interval = 10)
```

Arguments

job_name	The job name, as returned by cloud_check().
update_interval	The number of seconds between querying for updates

See Also

Other cloud: cloud_broken(), cloud_browse(), cloud_cancel(), cloud_check(), cloud_details(), cloud_fetch_results(), cloud_plot(), cloud_report(), cloud_results(), cloud_summary()

---

cloud_summary *Display revdep results*

Description

Displays nicely formatted results of processed packages run in the cloud.

Usage

```
cloud_summary(job_name = cloud_job(pkg = pkg), pkg = ".")
```

Arguments

job_name	The job name, as returned by cloud_check().
pkg	Path to package.

See Also

Other cloud: cloud_broken(), cloud_browse(), cloud_cancel(), cloud_check(), cloud_details(), cloud_fetch_results(), cloud_plot(), cloud_report(), cloud_results(), cloud_status()

---

cran_revdeps *Retrieve the reverse dependencies for a package*

Description

Retrieve the reverse dependencies for a package

Usage

```
cran_revdeps(package, dependencies = TRUE, bioc = FALSE, cran = TRUE)
```

Arguments

package	The package (or packages) to search for reverse dependencies.
dependencies	Which types of revdeps should be checked. For CRAN release, we recommend using the default.
bioc	Also check revdeps that live in Bioconductor?
cran	Should cran mirror be attached to getOpion("repos") if it is not already present.

---

revdep_add *Manage the package checking to-do list.*

Description

`revdep_todo()` tells you which packages still need to be checked. `revdep_add()` adds a single package to the to-do list. `revdep_rm()` removes packages from the todo list. `revdep_add_broken()` re-adds all broken packages from the last check (this is useful if you think you've fixed the underlying problem in your package).

Usage

```
revdep_add(pkg = ".", packages)

revdep_add_broken(
  pkg = ".",
  install_failures = FALSE,
  timeout_failures = FALSE
)

revdep_add_new(pkg = ".")

revdep_todo(pkg = ".")

revdep_rm(pkg = ".", packages)
```

Arguments

pkg	Path to package.
packages	Character vector of package names to add
install_failures	Whether to re-add packages that failed to install.
timeout_failures	Whether to re-add packages that timed out.

revdep_check *Run revdep checks*

Description

`revdep_check()` runs `R CMD check` on all reverse dependencies of your package. To avoid false positives, it runs `R CMD check` twice: once for released version on CRAN and once for the local development version. It then reports the differences so you can see what checks were previously ok but now fail.

It requires to use a repos option that provides the source code of the packages not binaries.

Once your package has been successfully submitted to CRAN, you should run `revdep_reset()`. This deletes all files used for checking, freeing up disk space and leaving you in a clean state for the next release.

Usage

```
revdep_check(
  pkg = ".",
  dependencies = c("Depends", "Imports", "Suggests", "LinkingTo"),
  quiet = TRUE,
  timeout = as.difftime(10, units = "mins"),
  num_workers = 1,
  bioc = TRUE,
  cran = TRUE,
  env = revdep_env_vars()
)

revdep_reset(pkg = ".")
```

Arguments

`pkg`	Path to package.
`dependencies`	Which types of revdeps should be checked. For CRAN release, we recommend using the default.
`quiet`	Suppress output from internal processes?
`timeout`	Maximum time to wait (in seconds) for `R CMD check` to complete. Default is 10 minutes.
`num_workers`	Number of parallel workers to use
`bioc`	Also check revdeps that live in Bioconductor?
`cran`	Should cran mirror be attached to getOpion("repos") if it is not already present.
`env`	Environment variables to set for the install and check processes. See `revdep_env_vars()`.

Details

`revdep_check()` proceeds in four steps:

1. **Init**: create the `revdep/` subdirectory if it doesn't already exist, and save the list of reverse dependencies to check.
2. **Install**: install the CRAN (released) and local (development) versions of your package, including all dependencies.
3. **Run**: run `R CMD check` twice for each reverse dependency, once for the CRAN version and one for the local version. The checks are run in parallel using `num_worker` processes.
4. **Report**: generate reports showing differences between the check results for the CRAN and local versions of your package. The focus of the report is on new failures. The reports are saved in `revdep/`.

`revdep_check()` is designed to seamlessly resume in the case of failure: just re-run `revdep_check()` and it will start from where it left off. If you want to start again from scratch, run `revdep_reset()`.

See Also

To see more details of problems during a run, call `revdep_summary()` and `revdep_details()` in another process.

`revdep_details` *Display revdep results*

Description

Use this to see nicely formatted results of processed packages while `revdep_check()` is running in another process. `revdep_summary()` displays summary results for all complete checks. `revdep_details()` shows you the details for one

Usage

```
revdep_details(pkg = ".", revdep)

revdep_summary(pkg = ".")
```

Arguments

`pkg`	Path to package
`revdep`	Name of revdep package.

`revdep_email` *Notify revdep maintainers about problems*

Description

This function uses gmail to automatically notify all maintainers of revdeps that have failures with the new version of the package. The form of the email is fixed, but it uses template parameters so that you can control the details: set the variables in `revdeps/email.yaml`. You'll be prompted to review the template before any emails are sent; or you can use `revdep_email_draft()` to see a draft version.

Usage

```
revdep_email(
  type = c("broken", "failed"),
  pkg = ".",
  packages = NULL,
  draft = FALSE
)

revdep_email_draft(type = "broken", pkg = ".", data = email_data(pkg))
```

Arguments

`type`	Type of problems to notify about; either "broken" (i.e. there is a new `R CMD check` failure that did not currently occur) or "failed" (i.e. the check failure either during installation or because of a timeout).
`pkg`	Path to package.
`packages`	A character vector of package names. Use this if some emails failed to send in the previous round. If omitted uses all packages.
`draft`	If `TRUE`, create a gmail draft rather than sending the email directly.
`data`	Optionally, supply a named list to provide your own parameters to fill in the template

Details

To use this function, you'll need to give the gmailr app authority to send emails from gmail. To revoke that authority, delete the `.httr-oauth` file created in your working directory.

revdep_env_vars *Environment variables to set for install and check processes while running the reverse dependency check*

Description

Environment variables to set for install and check processes while running the reverse dependency check

Usage

```
revdep_env_vars(force_suggests = FALSE)
```

Arguments

force_suggests	Whether to force the installation of the suggested packages.

Value

Named character vector.

revdep_maintainers *List maintainers of all reverse dependencies*

Description

List maintainers of all reverse dependencies

Usage

```
revdep_maintainers(pkg = ".")
```

Arguments

pkg	Path to package.

`revdep_report_summary` *Markdown report of reverse dependency check results*

Description

You can use these functions to get intermediate reports of a `revdep_check()` running in another session.

Usage

```
revdep_report_summary(pkg = ".", file = "", all = FALSE, results = NULL)

revdep_report_problems(
  pkg = ".",
  file = "",
  all = FALSE,
  results = NULL,
  bioc = TRUE,
  cran = TRUE
)

revdep_report_failures(
  pkg = ".",
  file = "",
  results = NULL,
  bioc = TRUE,
  cran = TRUE
)

revdep_report_cran(pkg = ".", file = "", results = NULL)

revdep_report(pkg = ".", all = FALSE, results = NULL, bioc = TRUE, cran = TRUE)
```

Arguments

`pkg`	Path to package.
`file`	File to write output to. Default will write to console.
`all`	Whether to report all problems, including the ones that were already present in the old version of the package. This potentially generated a lot of output, most of which was irrelevant, so they are omitted by default, and only problems seen with the new version of the package are reported.
`results`	Cached results from `db_results()`. Expert use only.
`bioc`	Also check revdeps that live in Bioconductor?
`cran`	Should cran mirror be attached to getOpion("repos") if it is not already present.

Details

`revdep_report_summary()` writes the contents of `README.md`, by default to the console. This is handy to quickly inspect the (current) list of problematic packages.

Index