Package 'rang'

July 23, 2025

Title Reconstructing Reproducible R Computational Environments

Version 0.3.0

Description Resolve the dependency graph of R packages at a specific time point based on the information from various 'R-hub' web services <https://blog.r-hub.io/>. The dependency graph can then be used to reconstruct the R computational environment with 'Rocker' <https://rocker-project.org>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.2.3

URL https://github.com/gesistsa/rang

BugReports https://github.com/gesistsa/rang/issues

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Imports parsedate, fastmap, jsonlite, memoise, pkgsearch, remotes, utils, httr, vctrs, renv, here

Depends R (>= 3.5.0)

VignetteBuilder knitr

LazyData true

NeedsCompilation no

Author Chung-hong Chan [aut, cre] (ORCID: <https://orcid.org/0000-0002-6232-7530>), David Schoch [aut] (ORCID: <https://orcid.org/0000-0003-2952-4812>), Egor Kotov [ctb] (ORCID: <https://orcid.org/0000-0001-6690-5345>)

Maintainer Chung-hong Chan <chainsawtiney@gmail.com>

Repository CRAN

Date/Publication 2023-10-08 14:50:02 UTC

Contents

`apptainerize` *Create an Apptainer/Singularity Definition File of The Resolved Result*

Description

This function exports the result from `resolve()` to an Apptainer/Singularity definition file. For R version >= 3.1.0, the file is based on the versioned Rocker Docker image. For R version < 3.1.0, the Apptainer/Singularity definition is based on Debian and it compiles R from source.

Usage

```
apptainerize(
  rang,
  output_dir,
  materials_dir = NULL,
  post_installation_steps = NULL,
  image = c("r-ver", "rstudio", "tidyverse", "verse", "geospatial"),
  rang_as_comment = TRUE,
  cache = FALSE,
  verbose = TRUE,
  lib = NA,
  cran_mirror = "https://cran.r-project.org/",
  check_cran_mirror = TRUE,
  bioc_mirror = "https://bioconductor.org/packages/",
  no_rocker = FALSE,
  debian_version = c("lenny", "squeeze", "wheezy", "jessie", "stretch"),
  skip_r17 = TRUE,
  insert_readme = TRUE,
  copy_all = FALSE
)
```

```
apptainerize_rang(...)

apptainerise(...)

apptainerise_rang(...)

singularize(...)

singularize_rang(...)

singularise(...)

singularise_rang(...)
```

Arguments

rang	output from resolve()
output_dir	character, where to put the Apptainer/Singularity definition file and associated content
materials_dir	character, path to the directory containing additional resources (e.g. analysis scripts) to be copied into output_dir and in turn into the Apptainer/Singularity container
post_installation_steps	character, additional steps to be added before the in the end of %post section the Apptainer/Singularity definition file, see an example below
image	character, which versioned Rocker image to use. Can only be "r-ver", "rstudio", "tidyverse", "verse", "geospatial" This applies only to R version >= 3.1
rang_as_comment	logical, whether to write resolved result and the steps to reproduce the file to path as comment
cache	logical, whether to cache the packages now. Please note that the system requirements are not cached. For query with non-CRAN packages, this option is strongly recommended. For query with local packages, this must be TRUE regardless of R version. For R version < 3.1, this must be also TRUE if there is any non-CRAN packages.
verbose	logical, pass to install.packages(), the negated value is also passed as quiet to both install.packages() and download.file().
lib	character, pass to install.packages(). By default, it is NA (to install the packages to the default location)
cran_mirror	character, which CRAN mirror to use
check_cran_mirror	logical, whether to check the CRAN mirror
bioc_mirror	character, which Bioconductor mirror to use
no_rocker	logical, whether to skip using Rocker images even when an appropriate version is available. Please keep this as TRUE unless you know what you are doing

debian_version	when Rocker images are not used, which EOL version of Debian to use. Can only be "lenny", "etch", "squeeze", "wheezy", "jessie", "stretch". Please keep this as default "lenny" unless you know what you are doing
skip_r17	logical, whether to skip R 1.7.x. Currently, it is not possible to compile R 1.7.x (R 1.7.0 and R 1.7.1) with the method provided by rang. It affects snapshot_date from 2003-04-16 to 2003-10-07. When skip_r17 is TRUE and snapshot_date is within the aforementioned range, R 1.8.0 is used instead
insert_readme	logical, whether to insert a README file
copy_all	logical, whether to copy everything in the current directory into the container. If inst/rang is detected in output_dir, this is coerced to TRUE.
...	arguments to be passed to apptainerize

Details

The idea behind this is to determine the installation order of R packages locally. Then, the installation script can be deployed to another fresh R session to install R packages. dockerize() and apptainerize() are more reasonable ways because a fresh R session with all system requirements is provided. The current approach does not work in R < 2.1.0.

Value

output_dir, invisibly

References

Apptainer / Singularity

Kurtzer, G. M., Sochat, V., & Bauer, M. W. (2017) Singularity: Scientific containers for mobility of compute. PLOS ONE, 12(5):e0177459. doi:10.1371/journal.pone.0177459

The Rocker Project

Ripley, B. (2005) Packages and their Management in R 2.1.0. R News, 5(1):8–11.

See Also

resolve(), export_rang(), use_rang()

Examples

```
if (interactive()) {
    graph <- resolve(
        pkgs = c("openNLP", "LDAvis", "topicmodels", "quanteda"),
        snapshot_date = "2020-01-16"
    )
    apptainerize(graph, ".")
    ## An example of using post_installation_steps to install quarto
    install_quarto <- c("apt-get install -y curl git && \\
    curl -LO https://quarto.org/download/latest/quarto-linux-amd64.deb && \\
    dpkg -i quarto-linux-amd64.deb && \\
    quarto install tool tinytex")
```

```
    apptainerize(graph, ".", post_installation_steps = install_quarto)
}
```

---

as_pkgrefs *Convert Data Structures into Package References*

Description

This generic function converts several standard data structures into a vector of package references, which in turn can be used as the first argument of the function `resolve()`. This function guessimates the possible sources of the packages. But we strongly recommend manually reviewing the detected packages before using them for `resolve()`.

Usage

```
as_pkgrefs(x, ...)

## Default S3 method:
as_pkgrefs(x, ...)

## S3 method for class 'character'
as_pkgrefs(x, bioc_version = NULL, no_enhances = TRUE, no_suggests = TRUE, ...)

## S3 method for class 'sessionInfo'
as_pkgrefs(x, ...)
```

Arguments

x	currently supported data structure(s) are: output from `sessionInfo()`, a character vector of package names
...	not used
bioc_version	character. When x is a character vector, version of Bioconductor to search for package names. NULL indicates not search for Bioconductor.
no_enhances	logical, when parsing DESCRIPTION, whether to ignore packages in the "Enhances" field
no_suggests	logical, when parsing DESCRIPTION, whether to ignore packages in the "Suggests" field

Value

a vector of package references

Examples

```
as_pkgrefs(sessionInfo())
if (interactive()) {
   require(rang)
   graph <- resolve(as_pkgrefs(sessionInfo()))
   as_pkgrefs(c("rtoot"))
   as_pkgrefs(c("rtoot", "S4Vectors")) ## this gives cran::S4Vectors and is not correct.
  as_pkgrefs(c("rtoot", "S4Vectors"), bioc_version = "3.3") ## This gives bioc::S4Vectors
}
```

`convert_edgelist` *Convert Data Structures to rang edgelist*

Description

This generic function converts several data structures provided by rang into an edgelist of package dependencies.

Usage

```
convert_edgelist(x, ...)

## Default S3 method:
convert_edgelist(x, ...)

## S3 method for class 'ranglet'
convert_edgelist(x, ...)

## S3 method for class 'rang'
convert_edgelist(x, ...)
```

Arguments

`x`	supported data structures are `rang` and `ranglet` S3 objects
`...`	not used

Details

the resulting data frame can be converted to an igraph object for plotting and analysis via the function `igraph::graph_from_data_frame()`

Value

a data frame of directed edges of dependencies

Examples

```
if (interactive()) {
    graph <- resolve(pkgs = c("openNLP", "LDAvis", "topicmodels", "quanteda"),
                snapshot_date = "2020-01-16")

    # dependency edgelist of a single package
    convert_edgelist(graph$ranglets[[1]])

    # full dependency edgelist
    convert_edgelist(graph)
}
```

`create_turing` *Create executable research compendium according to the Turing Way*

Description

This `usethis`-style function creates an executable research compendium according to the Turing Way.

Usage

```
create_turing(
  path,
  add_rang = TRUE,
  add_makefile = TRUE,
  add_here = TRUE,
  verbose = TRUE,
  force = FALSE,
  apptainer = FALSE
)
```

Arguments

`path`	character, path to the project root
`add_rang`	logical, whether to run `use_rang()` to `path`
`add_makefile`	logical, whether to insert a barebone `Makefile` in the project root.
`add_here`	logical, whether to insert a hidden `.here` file in the project root
`verbose`	logical, whether to print out messages
`force`	logical, whether to overwrite files (`inst/rang/update.R`, `Makefile`, `.here`) if they exist.
`apptainer`	logical, whether to use apptainer. `FALSE` indicates using Docker

Details

According to the Turing Way, an executable research compendium should have the following properties

1. Files should be organized in a conventional folder structure;
2. Data, methods, and output should be clearly separated;
3. The computational environment should be specified.

We use the structure suggested by the Turing Way:

- `data_raw`: a directory to hold the raw data
- `data_clean`: a directory to hold the processed data
- `code`: a directory to hold computer code
- `CITATION`: a file holding citation information
- `paper.Rmd`: a manuscript This function provides the a clearly separated organizational structure. Components can be changed. For example, the manuscript can be in another format (e.g. quarto, sweave) or even optional. With `add_rang`, the computational environment can be recorded and reconstructed later.

Value

path, invisibly

References

The Turing Way: Research Compendia Gorman, KB, Williams TD. and Fraser WR (2014). Ecological Sexual Dimorphism and Environmental Variability within a Community of Antarctic Penguins (Genus Pygoscelis). PLoS ONE 9(3):e90081. doi:10.1371/journal.pone.0090081

See Also

`use_rang()`

`dockerize` *Dockerize The Resolved Result*

Description

This function exports the result from `resolve()` to a Docker file. For R version >= 3.1.0, the Dockerfile is based on the versioned Rocker image. For R version < 3.1.0, the Dockerfile is based on Debian and it compiles R from source.

Usage

```
dockerize(
  rang,
  output_dir,
  materials_dir = NULL,
  post_installation_steps = NULL,
  image = c("r-ver", "rstudio", "tidyverse", "verse", "geospatial"),
  rang_as_comment = TRUE,
  cache = FALSE,
  verbose = TRUE,
  lib = NA,
  cran_mirror = "https://cran.r-project.org/",
  check_cran_mirror = TRUE,
  bioc_mirror = "https://bioconductor.org/packages/",
  no_rocker = FALSE,
  debian_version = c("lenny", "squeeze", "wheezy", "jessie", "stretch"),
  skip_r17 = TRUE,
  insert_readme = TRUE,
  copy_all = FALSE
)

dockerize_rang(...)

dockerise(...)

dockerise_rang(...)
```

Arguments

rang	output from resolve()
output_dir	character, where to put the Docker file and associated content
materials_dir	character, path to the directory containing additional resources (e.g. analysis scripts) to be copied into output_dir and in turn into the Docker container
post_installation_steps	character, additional steps to be added before the CMD part of the Dockerfile, see an example below
image	character, which versioned Rocker image to use. Can only be "r-ver", "rstudio", "tidyverse", "verse", "geospatial" This applies only to R version >= 3.1
rang_as_comment	logical, whether to write resolved result and the steps to reproduce the file to path as comment
cache	logical, whether to cache the packages now. Please note that the system requirements are not cached. For query with non-CRAN packages, this option is strongly recommended. For query with local packages, this must be TRUE regardless of R version. For R version < 3.1, this must be also TRUE if there is any non-CRAN packages.

verbose	logical, pass to install.packages(), the negated value is also passed as quiet to both install.packages() and download.file().
lib	character, pass to install.packages(). By default, it is NA (to install the packages to the default location)
cran_mirror	character, which CRAN mirror to use
check_cran_mirror	logical, whether to check the CRAN mirror
bioc_mirror	character, which Bioconductor mirror to use
no_rocker	logical, whether to skip using Rocker images even when an appropriate version is available. Please keep this as TRUE unless you know what you are doing
debian_version	when Rocker images are not used, which EOL version of Debian to use. Can only be "lenny", "etch", "squeeze", "wheezy", "jessie", "stretch". Please keep this as default "lenny" unless you know what you are doing
skip_r17	logical, whether to skip R 1.7.x. Currently, it is not possible to compile R 1.7.x (R 1.7.0 and R 1.7.1) with the method provided by rang. It affects snapshot_date from 2003-04-16 to 2003-10-07. When skip_r17 is TRUE and snapshot_date is within the aforementioned range, R 1.8.0 is used instead
insert_readme	logical, whether to insert a README file
copy_all	logical, whether to copy everything in the current directory into the container. If inst/rang is detected in output_dir, this is coerced to TRUE.
...	arguments to be passed to dockerize

Details

The idea behind this is to determine the installation order of R packages locally. Then, the installation script can be deployed to another fresh R session to install R packages. dockerize() and apptainerize() are more reasonable ways because a fresh R session with all system requirements is provided. The current approach does not work in R < 2.1.0.

Value

output_dir, invisibly

References

The Rocker Project Ripley, B. (2005) Packages and their Management in R 2.1.0. R News, 5(1):8–11.

See Also

resolve(), export_rang(), use_rang()

Examples

```
if (interactive()) {
    graph <- resolve(pkgs = c("openNLP", "LDAvis", "topicmodels", "quanteda"),
                     snapshot_date = "2020-01-16")
    dockerize(graph, ".")
    ## An example of using post_installation_steps to install quarto
    install_quarto <- c("RUN apt-get install -y curl git && \\
    curl -LO https://quarto.org/download/latest/quarto-linux-amd64.deb && \\
    dpkg -i quarto-linux-amd64.deb && \\
    quarto install tool tinytex")
    dockerize(graph, ".", post_installation_steps = install_quarto)
}
```

`export_rang` *Export The Resolved Result As Installation Script*

Description

This function exports the results from `resolve()` to an installation script that can be run in a fresh R environment.

Usage

```
export_rang(
  rang,
  path,
  rang_as_comment = TRUE,
  verbose = TRUE,
  lib = NA,
  cran_mirror = "https://cran.r-project.org/",
  check_cran_mirror = TRUE,
  bioc_mirror = "https://bioconductor.org/packages/"
)
```

Arguments

`rang`	output from `resolve()`
`path`	character, path of the exported installation script
`rang_as_comment`	logical, whether to write resolved result and the steps to reproduce the file to `path` as comment
`verbose`	logical, pass to `install.packages()`, the negated value is also passed as `quiet` to both `install.packages()` and `download.file()`.
`lib`	character, pass to `install.packages()`. By default, it is NA (to install the packages to the default location)

cran_mirror	character, which CRAN mirror to use
check_cran_mirror	logical, whether to check the CRAN mirror
bioc_mirror	character, which Bioconductor mirror to use

Details

The idea behind this is to determine the installation order of R packages locally. Then, the installation script can be deployed to another fresh R session to install R packages. dockerize() and apptainerize() are more reasonable ways because a fresh R session with all system requirements is provided. The current approach does not work in R < 2.1.0.

Value

path, invisibly

References

Ripley, B. (2005) Packages and their Management in R 2.1.0. R News, 5(1):8–11.

See Also

generate_installation_order()

Examples

```
if (interactive()) {
    graph <- resolve(pkgs = c("openNLP", "LDAvis", "topicmodels", "quanteda"),
                     snapshot_date = "2020-01-16")
    export_rang(graph, "rang.R")
}
```

---

export_renv *Export The Resolved Result As a renv Lockfile*

Description

This function exports the results from resolve() to a renv lockfile that can be used as an alternative to a docker container.

Usage

```
export_renv(rang, path = ".")
```

Arguments

rang	output from resolve()
path	character, path of the exported renv lockfile

Details

A renv lockfile is easier to handle than a docker container, but it cannot always reliably reproduce the exact computational environment,especially for very old code.

Value

`path`, invisibly

Examples

```
if (interactive()) {
    graph <- resolve(pkgs = c("openNLP", "LDAvis", "topicmodels", "quanteda"),
                     snapshot_date = "2020-01-16")
    export_renv(graph, ".")
}
```

`generate_installation_order`

Create a Data Frame of The Resolved Result This function exports the results from `resolve()` to a data frame, which each row represents one installation step. The order of rows is the installation order. By installing packages in the specified order, one can install all the resolved packages without conflicts.

Description

Create a Data Frame of The Resolved Result This function exports the results from `resolve()` to a data frame, which each row represents one installation step. The order of rows is the installation order. By installing packages in the specified order, one can install all the resolved packages without conflicts.

Usage

```
generate_installation_order(rang)
```

Arguments

`rang`	output from `resolve()`

Value

A data frame ordered by installation order.

References

Ripley, B. (2005) Packages and their Management in R 2.1.0. R News, 5(1):8–11.

Examples

```
if (interactive()) {
    graph <- resolve(pkgs = c("openNLP", "LDAvis", "topicmodels", "quanteda"),
                     snapshot_date = "2020-01-16")
    generate_installation_order(graph)
}
```

`query_sysreqs` *Query for System Requirements*

Description

This function takes an S3 object returned from `resolve()` and (re)queries the System Requirements.

Usage

```
query_sysreqs(rang, os = "ubuntu-20.04")
```

Arguments

`rang`	output from `resolve()`
`os`	character, which OS to query for system requirements

Value

a `rang` S3 object with the following items

`call`	original function call
`ranglets`	List of dependency graphs of all packages in `pkgs`
`snapshot_date`	`snapshot_date`
`no_enhances`	`no_enhances`
`no_suggests`	`no_suggests`
`unresolved_pkgsrefs`	Packages that can't be resolved
`sysreqs`	System requirements as Linux commands
`r_version`	The latest R version as of `snapshot_date`
`os`	`os`

See Also

`resolve()`

Examples

```
if (interactive()) {
    graph <- resolve(pkgs = c("openNLP", "LDAvis", "topicmodels", "quanteda"),
                snapshot_date = "2020-01-16", query_sysreqs = FALSE)
    graph$sysreqs
    graph2 <- query_sysreqs(graph, os = "ubuntu-20.04")
    graph2$sysreqs
}
```

---

recipes *Recipes for Building Container Images*

Description

A list containing several useful recipes for container building. Useful for the `post_installation_steps` argument of `dockerize()`. Available recipes are:

- `texlive`: install pandoc and LaTeX, useful for rendering RMarkdown
- `texlivefull`: Similar to the above, but install the full distribution of TeX Live (~ 3GB)
- `quarto`: install quarto and tinytex
- `clean`: clean up the container image by removing cache
- `make`: install GNU make

Usage

```
recipes
```

Format

An object of class `list` of length 5.

Examples

```
if (interactive()) {
    graph <- resolve(pkgs = c("openNLP", "LDAvis", "topicmodels", "quanteda"),
                     snapshot_date = "2020-01-16")
    ## install texlive
    dockerize(graph, ".", post_installation_steps = recipes[['texlive']])
}
```

`resolve` *Resolve Dependencies Of R Packages*

Description

This function recursively queries dependencies of R packages at a specific snapshot time. The dependency graph can then be used to recreate the computational environment. The data on dependencies are provided by R-hub.

Usage

```
resolve(
  pkgs = ".",
  snapshot_date,
  no_enhances = TRUE,
  no_suggests = TRUE,
  query_sysreqs = TRUE,
  os = "ubuntu-20.04",
  verbose = FALSE
)
```

Arguments

`pkgs`	`pkgs` can be 1) a character vector of R packages to resolve, 2) a path to a renv lockfile, or 3) a data structure that `as_pkgrefs()` can convert to a character vector of package references. For 1) `pkgs` can be either in shorthands, e.g. "rtoot", "ropensci/readODS", or in package references, e.g. "cran::rtoot", "github::ropensci/readODS". Please refer to the Package References documentation of pak for details. Currently, this package supports only cran and github packages. For 2) `as_pkgrefs()` support the output of `sessionInfo()`, a renv lockfile or a single directory. If it is a single directory, all R scripts are scanned for R packages used using `renv::dependencies()`. Currently, the default is to scan the R scripts in the current working directory. Please also note that this scanning only assumes there are CRAN and Bioconductor packages. We strongly recommend checking whether this is really the case (see example below).
`snapshot_date`	Snapshot date, if not specified, assume to be a month ago
`no_enhances`	logical, whether to ignore packages in the "Enhances" field
`no_suggests`	logical, whether to ignore packages in the "Suggests" field
`query_sysreqs`	logical, whether to query for System Requirements. Important: Archived CRAN can't be queried for system requirements. Those packages are assumed to have no system requirement.
`os`	character, which OS to query for system requirements
`verbose`	logical, whether to display messages

Value

a rang S3 object with the following items

call	original function call
ranglets	List of dependency graphs of all packages in pkgs
snapshot_date	snapshot_date
no_enhances	no_enhances
no_suggests	no_suggests
unresolved_pkgsrefs	Packages that can't be resolved
sysreqs	System requirements as Linux commands
r_version	The latest R version as of snapshot_date
os	os

References

Package References

See Also

dockerize()

Examples

```
if (interactive()) {
    graph <- resolve(pkgs = c("openNLP", "LDAvis", "topicmodels", "quanteda"),
                snapshot_date = "2020-01-16")
    graph
    ## to resolve github packages
    gh_graph <- resolve(pkgs = c("https://github.com/schochastics/rtoot"),
                    snapshot_date = "2022-11-28")
    gh_graph
    ## scanning
    graph <- resolve(snapshot_date = "2022-11-28")
    ## But we recommend this:
    pkgs <- as_pkgrefs(".")
    pkgs ## check the accuracy
    graph <- resolve(pkgs, snapshot_date = "2022-11-28")
}
```

use_rang *Setup rang for a directory*

Description

This usethis-style function adds the infrastructure in a directory (presumably with R scripts and data) for (re)constructing the computational environment. Specifically, this function inserts inst/rang into the directory, which contains all components for the reconstruction. Optionally, Makefile and .here are also inserted to ease the development of analytic code. By default, (re)running this function does not overwrite any file. One can change this by setting force to TRUE.

Usage

```
use_rang(
  path = ".",
  add_makefile = TRUE,
  add_here = TRUE,
  verbose = TRUE,
  force = FALSE,
  apptainer = FALSE
)
```

Arguments

path	character, path to the project root
add_makefile	logical, whether to insert a barebone Makefile in the project root.
add_here	logical, whether to insert a hidden .here file in the project root
verbose	logical, whether to print out messages
force	logical, whether to overwrite files (inst/rang/update.R, Makefile, .here) if they exist.
apptainer	logical, whether to use apptainer. FALSE indicates using Docker

Details

The infrastructure being added to your path consists of:

- inst/rang directory in the project root
- update.R file inside the directory
- .here in the project root (if add_here is TRUE)
- Makefile in the project root (if add_makefile is TRUE) You might need to edit update.R manually. The default is to scan the whole project for used R packages and assume they are either on CRAN or Bioconductor. If you have used other R packages, you might need to edit this manually.

Value

path, invisibly

Index