Package 'bigbang'

August 8, 2026

Type Package

Title Build 'Tidyverse'-Style Meta-Packages from Local Package Files

Version 0.1.0

Description Creates meta-packages (in the style of the 'tidyverse') from locally stored package archives (.tar.gz, .zip). Designed for teams working behind institutional firewalls without access to package repositories. Resolves dependencies between local packages by building a dependency graph with topological ordering and cycle detection, classifies dependencies as local or CRAN, detects implicit dependencies by scanning source code, and generates the full meta-package scaffold including re-exports, vignettes and documentation.

License GPL (>= 3)

URL https://github.com/sebollin/bigbang

BugReports https://github.com/sebollin/bigbang/issues

Encoding UTF-8

Language en

VignetteBuilder knitr

Depends R (>= 3.6.0)

Imports brio, glue, utils, whisker

Suggests devtools, knitr, rmarkdown, testthat (>= 3.0.0), withr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Sebastian Lucas [aut, cre],
Richard Detomasi [ctb] (ORCID: <https://orcid.org/0000-0002-6725-0261>,
Suggested the initial idea of a metapackage builder)

Maintainer Sebastian Lucas <sebalucas@gmail.com>

Repository CRAN

Date/Publication 2026-08-08 12:50:11 UTC

Contents

`create_metapackage` *Build a local meta-package*

Description

Creates the full structure and files of a meta-package that installs, manages and loads a set of locally stored R packages, resolving the dependencies between them with a graph-based (topologically ordered) approach.

Usage

```
create_metapackage(
  name,
  packages,
  pkg_dir,
  ext = ".tar.gz",
  version = "0.1.0",
  dest_dir,
  reexport = FALSE,
  document = TRUE,
  verbose = getOption("bigbang.verbose", interactive()),
  authors =
   "person('First', 'Last', email = 'first.last@example.com', role = c('aut', 'cre'))",
  description = "Local Package Metapackage",
  license = "MIT + file LICENSE",
  additional_deps = NULL,
  ignore_deps = NULL,
 import_deps = c("data.table", "dplyr", "ggplot2", "readr", "tibble", "tidyr", "xts",
    "zoo"),
  force_deps = NULL,
  debug = FALSE
)
```

Arguments

name	Character. Name of the meta-package to create (must not contain underscores _).
packages	Character vector. Names (with version) of the local packages to include, e.g. "myPackage_1.0.0".
pkg_dir	Character. Directory containing the local archive files (.tar.gz, .zip, etc.).
ext	Character. Archive extension. Defaults to ".tar.gz".
version	Character. Version of the meta-package. Defaults to "0.1.0".
dest_dir	Character. Required destination directory. The function writes the generated meta-package exclusively inside this directory; there is no default path. Use tempdir() for disposable output.
reexport	Logical. If TRUE, re-exports the component packages' functions so they are reachable directly through the meta-package (tidyverse style). Defaults to FALSE.
document	Logical. If TRUE, runs devtools::document() automatically. Defaults to TRUE.
verbose	Logical. If TRUE, shows verbose messages. The default follows getOption("bigbang.verbose", interactive()).
authors	Character. Content for the Authors@R field of DESCRIPTION.
description	Character. Description of the meta-package.
license	Character. License of the meta-package.
additional_deps	Character vector. Extra dependencies to add on top of the ones detected automatically.
ignore_deps	Character vector. Dependencies to ignore even if detected.
import_deps	Character vector. Packages that should go in the Imports field of DESCRIPTION rather than Depends. Imports are not attached when the user calls library() on the meta-package, but remain available via :: (e.g. dplyr::filter()), reducing name clashes in the user's workspace.
force_deps	Character vector. Exact package names to use as dependencies, bypassing automatic detection. If supplied, only these are used as the meta-package's implicit dependencies.
debug	Logical. If TRUE, emits detailed debugging messages. Defaults to FALSE.

Details

The function performs the following steps:

1. Creates the basic R package structure (R, man, vignettes, etc.).
2. Detects dependencies between packages, both explicit (from DESCRIPTION) and implicit (found by scanning the source code).
3. Generates DESCRIPTION and NAMESPACE with the appropriate dependencies.
4. Creates a basic vignette documenting the meta-package.
5. Generates R files with functions to install and load the component packages:

- <name>_install(): installs the component packages from the local archives.
- <name>_attach(): attaches the components that are already installed.
- <name>_detach(): detaches all the meta-package's components.
- <name>_packages(): lists the included packages.

Installation is **explicit**: calling library(<meta>) attaches the components that are already installed and reports which ones are missing, but does not install anything or delete any files. To install the components from the local archives, the user calls <meta>_install(). Installation resolves dependencies with a graph-based topological ordering that also detects circular dependencies.

Value

Invisibly, a bigbang_result containing the generated path, component archives, dependency classification, and documentation status.

Component installation

The generated meta-package installs component packages only when the user explicitly calls <meta>_install(). Loading it with library() never installs packages. By default, the generated installer does not access a repository.

If reexport = TRUE, a reexports.R file is generated so users can reach the component functions directly through the meta-package (meta::fun() instead of component::fun()), tidyverse style.

Requirements

- The local packages must exist in pkg_dir with the given extension.
- Automatic documentation (document = TRUE) requires the devtools package.

Examples

```
archives <- system.file("extdata", package = "bigbang")
destination <- tempfile("bigbang-example-")
dir.create(destination)

result <- create_metapackage(
  name = "toyverse",
  packages = "toycomponent_0.1.0",
  pkg_dir = archives,
  dest_dir = destination,
  document = FALSE,
  verbose = FALSE,
  import_deps = character(),
  force_deps = character()
)
list.files(result$path)

unlink(destination, recursive = TRUE)
```

`diagnose_dependencies` *Diagnose implicit dependencies of local packages*

Description

Scans local packages for references to the recommended packages 'Matrix' and 'class', which can cause `R CMD check` failures when they are used implicitly but not declared as dependencies.

Usage

```
diagnose_dependencies(packages, pkg_dir, ext = ".tar.gz")
```

Arguments

`packages`	Character vector. Names (with version) of the local packages to examine, e.g. `"conexiones_0.8.3"`.
`pkg_dir`	Character. Directory containing the local archive files (`.tar.gz`, `.zip`, etc.).
`ext`	Character. Archive extension. Defaults to `".tar.gz"`.

Details

Extracts and scans the R source of each package for patterns that suggest implicit use of 'Matrix' or 'class'. Useful for debugging `R CMD check` errors such as "there is no package called 'Matrix'" even when the package does not appear to use it directly.

Value

A named list with one entry per local package, each a list with two elements:

matrix_refs Character vector of references to 'Matrix', with file and line.

class_refs Character vector of references to 'class', with file and line.

Examples

```
archives <- system.file("extdata", package = "bigbang")
res <- diagnose_dependencies(
  packages = "toycomponent_0.1.0",
  pkg_dir = archives
)
res[["toycomponent_0.1.0"]]
lapply(res, function(x) x$matrix_refs)
```

`install_local_pkg` *Install a local package together with its dependencies*

Description

Installs a package from a local archive. Dependencies available as local archives are installed recursively; missing non-local dependencies follow the explicit `cran_deps` policy. ZIP archives containing `Meta/package.rds` are treated as Windows binaries, while other ZIP archives are unpacked and installed as source packages.

Usage

```
install_local_pkg(
  package,
  pkg_dir,
  ext = ".tar.gz",
  repos = getOption("repos"),
  cran_deps = c("skip", "error", "install"),
  verbose = getOption("bigbang.verbose", interactive())
)
```

Arguments

`package`	Character. Package file name without extension (for example, `"uspr_0.8.5"`).
`pkg_dir`	Character. Directory containing local archives.
`ext`	Character. Archive extension.
`repos`	Character. Repositories used only when `cran_deps = "install"`.
`cran_deps`	Character. Policy for missing non-local dependencies: `"skip"` (the default) never accesses the network, `"error"` fails without accessing it, and `"install"` attempts installation from `repos`.
`verbose`	Logical. Whether to emit progress and summary messages. The default follows `getOption("bigbang.verbose", interactive())`.

Value

Invisibly, a list describing installed, failed, and skipped packages.

Installation

This function installs packages into the user's active R library. Installation occurs only when the user calls the function; loading `bigbang` never installs packages. With the default `cran_deps = "skip"`, it does not access the network.

See Also

`create_metapackage()` for generating a meta-package with an explicit component installer.

print.bigbang_artifact_scan *Print an artifact scan*

Description

Print an artifact scan

Usage

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

Arguments

x	A bigbang_artifact_scan object.
...	Unused.

Value

x, invisibly.

print.bigbang_install_result *Print a local package installation result*

Description

Print a local package installation result

Usage

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

Arguments

x	A bigbang_install_result object.
...	Unused.

Value

x, invisibly.

print.bigbang_result *Print a metapackage generation result*

Description

Print a metapackage generation result

Usage

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

Arguments

x	A bigbang_result object returned by create_metapackage().
...	Unused.

Value

x, invisibly.

scan_bigbang_artifact *Scan a generated metapackage for historical deletion signatures*

Description

Inspects a generated metapackage source directory, source archive, or installed package without loading it. The scanner looks for the historical V1, V2, V3, and V7 deletion signatures from the pre-release security investigation. Provenance is read from both the current generator fields and the legacy pre-rename fields so development artifacts remain classifiable.

Usage

```
scan_bigbang_artifact(path, dry_run = TRUE)
```

Arguments

path	Character scalar. Source directory, .tar.gz/.tar/.zip source archive, or installed package directory.
dry_run	Logical. Must be TRUE, the default. Automatic mutation or remediation is deliberately not implemented.

Details

Installed packages are inspected through R's internal lazy-load database API. That code is isolated in .scan_installed_lazydb() and has been exercised with R 4.6.1. Because this is an internal R format, callers should re-run the scanner tests when adopting a new R minor release.

Value

A list with the artifact type, vulnerability flag, detected signatures, evidence locations, provenance fields, and R version used for the scan.

Examples

```
archives <- system.file("extdata", package = "bigbang")
destination <- tempfile("bigbang-scan-example-")
dir.create(destination)

result <- create_metapackage(
  name = "toyverse",
  packages = "toycomponent_0.1.0",
  pkg_dir = archives,
  dest_dir = destination,
  document = FALSE,
  verbose = FALSE,
  import_deps = character(),
  force_deps = character()
)
scan_bigbang_artifact(result$path)

unlink(destination, recursive = TRUE)
```

Index