

# Package: binman (via r-universe)

October 28, 2024

**Title** A Binary Download Manager

**Version** 0.1.3

**Description** Tools and functions for managing the download of binary files. Binary repositories are defined in 'YAML' format. Defining new pre-download, download and post-download templates allow additional repositories to be added.

**License** MIT + file LICENSE

**URL** <https://docs.ropensci.org/binman/>,  
<https://github.com/ropensci/binman>

**BugReports** <https://github.com/ropensci/binman/issues>

**Depends** R (>= 3.3)

**Imports** assertthat, httr, jsonlite, rappdirs, semver, stats, utils, xml2, yaml

**Suggests** covr, knitr, rmarkdown, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**Repository** <https://ropensci.r-universe.dev>

**RemoteUrl** <https://github.com/ropensci/binman>

**RemoteRef** master

**RemoteSha** 8149b1ea3b36f6e73ee81e6d0a7a14ec048a75bc

## Contents

app_dir	2
assign_directory	3
binman	3
download_files	4

list_versions . . . . .	5
noproc_dlfiles . . . . .	5
predl_bitbucket_downloads . . . . .	6
predl_github_assets . . . . .	7
predl_google_storage . . . . .	8
process_yaml . . . . .	10
rm_platform . . . . .	11
rm_version . . . . .	11
unziptar_dlfiles . . . . .	12

<b>Index</b>	<b>14</b>
--------------	-----------

---

app_dir	<i>Get application directory</i>
---------	----------------------------------

---

### Description

Get application directory

### Usage

```
app_dir(appname, check = TRUE)
```

### Arguments

appname	A character string giving the name of the application
check	check whether the app given by appname exists or not.

### Value

A character string giving the path of the directory

### Examples

```
## Not run:
appdir <- app_dir("superduperapp", FALSE)

## End(Not run)
```

---

assign_directory	<i>Assign directory</i>
------------------	-------------------------

---

**Description**

Assign directory to download list

**Usage**

```
assign_directory(dllist, appname)
```

**Arguments**

dllist	A named list of data.frames. The name indicates the platform. The data.frame should contain the version, url and file to be processed.
appname	Name to give the app

**Value**

A named list of data.frames. The data.frame should contain the version, url and file to be processed, the directory to download the file to and whether the file already exists.

**Examples**

```
## Not run:
tdata <- system.file("testdata", "test_dllist.Rdata", package = "binman")
load(tdata)
assign_directory(test_dllist, "myapp")

## End(Not run)
```

---

binman	<i>binman</i>
--------	---------------

---

**Description**

A Binary Download Manager.

**Details**

Tools and functions for managing the download of binary files. Binary repositories are defined in 'YAML' format. Defining new pre-download, download and post-download templates allow additional repositories to be added.

---

download_files	<i>Download binaries</i>
----------------	--------------------------

---

### Description

Download binaries from repository

### Usage

```
download_files(dllist, overwrite = FALSE)
```

### Arguments

dllist	A named list of data.frames. The data.frame should contain the version, url and file to be processed, the directory to download the file to and whether the file already exists.
overwrite	Overwrite existing binaries. Default value of FALSE

### Value

A data.frame indicating whether a file was downloaded for a platform.

### Examples

```
## Not run:
trdata <- system.file("testdata", "test_dlres.Rdata", package = "binman")
tlldata <- system.file("testdata", "test_dllist.Rdata", package = "binman")
load(trdata)
load(tlldata)
dllist <- assign_directory(test_dllist, "myapp")
testthat::with_mock(
  `httr::GET` = function(...) {
    test_llres
  },
  `base::dir.create` = function(...) {
    TRUE
  },
  dlfiles <- download_files(dllist)
)

## End(Not run)
```

---

list_versions	<i>List app versions</i>
---------------	--------------------------

---

**Description**

List app versions by platform

**Usage**

```
list_versions(appname, platform = c("ALL"))
```

**Arguments**

appname	A character string giving the name of the application
platform	A character vector of platforms to list. Defaults to "ALL"

**Value**

A list of platforms with version directories

**Examples**

```
## Not run:
appdir <- app_dir("superduperapp", FALSE)
platforms <- LETTERS[1:4]
versions <- LETTERS[5:7]
makedirs <- file.path(appdir, outer(platforms, versions, file.path))
chk <- vapply(makedirs, dir.create, logical(1), recursive = TRUE)
expect_true(all(chk))
res <- list_versions("superduperapp")
unlink(appdir, recursive = TRUE)

## End(Not run)
```

---

noproc_dlfiles	<i>Do not post process</i>
----------------	----------------------------

---

**Description**

Do not post process dlfiles

**Usage**

```
noproc_dlfiles(dlfiles)
```

**Arguments**

dlfiles            A data.frame of files by platform and indicating whether they were processed

**Value**

Returns a list of character vectors indicating files processed

**Examples**

```
## Not run:
ymlfile <- system.file("exdata", "sampleapp4.yml", package = "binman")
trdata <- system.file("testdata", "test_dlres.Rdata", package = "binman")
load(trdata)
testthat::with_mock(
  `htr::GET` = function(...) {
    test_llres
  },
  `base::dir.create` = function(...) {
    TRUE
  },
  procyaml <- process_yaml(ymlfile)
)
procyaml

## End(Not run)
```

---

predl\_bitbucket\_downloads

*Pre download bitbucket downloads*

---

**Description**

Pre download bitbucket downloads template function

**Usage**

```
predl_bitbucket_downloads(
  url,
  platform,
  history,
  appname,
  platformregex = platform,
  versionregex = "\\d+(?:\\.\\d+)+"
)
```

## Arguments

url	A url giving the bitbucket download JSON for a project. As an example <a href="https://bitbucket.org/ariya/phantomjs">https://bitbucket.org/ariya/phantomjs</a> the phantomjs project has an asset JSON available at <a href="https://api.bitbucket.org/2.0/repositories/ariya/phantomjs">https://api.bitbucket.org/2.0/repositories/ariya/phantomjs</a>
platform	A character vector of platform names
history	The maximum number of files to get for a platform
appname	Name of the app
platformregex	A filter for platforms. Defaults to the platform
versionregex	A regex for retrieving the version.

## Value

A named list of data.frames. The name indicates the platform. The data.frame should contain the version, url and file to be processed. Used as input for [download\\_files](#) or an equivalent.

## Examples

```
## Not run:
bbdata <- system.file("testdata", "test_bitbucketdl.json",
  package = "binman"
)
platform <- c("linux64", "linux32", "windows", "macosx")
platformregex <- c("linux-x86_64", "linux-i686", "windows", "macosx")
bbdlist <-
  predl_bitbucket_downloads(
    url = bbdata, platform, history = 3L,
    appname = "binman_chromedriver",
    platformregex
  )

## End(Not run)
```

---

predl\_github\_assets *Pre download Github assets*

---

## Description

Pre download Github assets template function

## Usage

```
predl_github_assets(
  url,
  platform,
  history,
  appname,
  fileregex = "",
```

```

platformregex = platform,
versionregex = c("", "")
)

```

### Arguments

url	A url giving the github asset JSON for a project. As an example <a href="https://github.com/mozilla/geckodriver/releases">https://github.com/mozilla/geckodriver/releases</a> the geckodriver project has an asset JSON available at <a href="https://api.github.com/repos/mozilla/geckodriver/releases">https://api.github.com/repos/mozilla/geckodriver/releases</a>
platform	A character vector of platform names
history	The maximum number of files to get for a platform
appname	Name of the app
fileregex	A filter for files
platformregex	A filter for platforms. Defaults to the platform
versionregex	A regex for retrieving the version.

### Value

A named list of data.frames. The name indicates the platform. The data.frame should contain the version, url and file to be processed. Used as input for [download\\_files](#) or an equivalent.

### Examples

```

## Not run:
gadata <- system.file("testdata", "test_gitassets.json",
  package = "binman"
)
platform <- c("linux64", "win64", "macos")
gadllist <- predl_github_assets(
  url = gadata, platform, history = 3L,
  appname = "binman_chromedriver"
)

## End(Not run)

```

---

predl\_google\_storage *Pre-Download Google Storage*

---

### Description

Pre-Download Google Storage template function

**Usage**

```
predl_google_storage(
  url,
  platform,
  history,
  appname,
  fileregex = "\\\\.zip$",
  platformregex = platform,
  versionregex = c(paste0("(.*)/.*", fileregex), "\\1")
)
```

**Arguments**

url	A url giving the JSON bucket listings for a project. For example: <a href="http://chromedriver.storage.googleapis.com">http://chromedriver.storage.googleapis.com</a> lists the chromedriver files but <a href="https://www.googleapis.com/storage/v1/b/chromedriver/o/">https://www.googleapis.com/storage/v1/b/chromedriver/o/</a> is the JSON listings for the project.
platform	A character vector of platform names
history	The maximum number of files to get for a platform
appname	Name of the app
fileregex	A filter for files
platformregex	A filter for platforms. Defaults to the platform names.
versionregex	A regex for retrieving the version.

**Value**

A named list of data.frames. The name indicates the platform. The data.frame should contain the version, url and file to be processed. Used as input for [download\\_files](#) or an equivalent.

**Examples**

```
## Not run:
gsdata <- system.file("testdata", "test_googstor.json",
  package = "binman"
)
platform <- c("linux64", "win32", "mac64")
gsdllist <- predl_google_storage(
  url = gsdata, platform, history = 5L,
  appname = "binman_chromedriver"
)

## End(Not run)
```

---

process_yaml	<i>Process a yaml file</i>
--------------	----------------------------

---

### Description

Process a yaml file. The file defines the pre-download function, the download function and the post download function.

### Usage

```
process_yaml(ymlfile, verbose = TRUE)
```

### Arguments

ymlfile	A file in a YAML format defining the pre-download/ download and post download functions together with their arguments.
verbose	If TRUE, include status messages (if any)

### Value

A list of files processed (downloaded and post processed)

### Examples

```
## Not run:
ymlfile <- system.file("exdata", "sampleapp.yaml", package = "binman")
trdata <- system.file("testdata", "test_dlres.Rdata", package = "binman")
load(trdata)
testthat::with_mock(
  `httr::GET` = function(...) {
    test_llres
  },
  `base::dir.create` = function(...) {
    TRUE
  },
  `utils::unzip` = function(zipfile, ...) {
    zipfile
  },
  procyaml <- process_yaml(ymlfile)
)
procyaml

## End(Not run)
```

---

rm_platform	<i>Remove application platform</i>
-------------	------------------------------------

---

**Description**

Remove application files/directories for a given platform

**Usage**

```
rm_platform(appname, platform = c("ALL"))
```

**Arguments**

appname	A character string giving the name of the application
platform	A character vector indicating the platform to remove. Defaults to "ALL"

**Value**

Returns a logical vector indicating whether the removal of platform was successful. Return is invisible.

**Examples**

```
## Not run:
appdir <- app_dir(appname, FALSE)
platforms <- LETTERS[1:4]
versions <- LETTERS[5:7]
makedirs <- file.path(appdir, outer(platforms, versions, file.path))
chk <- vapply(makedirs, dir.create, logical(1), recursive = TRUE)
appver <- list_versions(appname)
rm_platform(appname, platforms[2:3])
unlink(appdir, recursive = TRUE)

## End(Not run)
```

---

rm_version	<i>Remove application version</i>
------------	-----------------------------------

---

**Description**

Remove application version for a given platform

**Usage**

```
rm_version(appname, platform, version = c("ALL"))
```

**Arguments**

appname	A character string giving the name of the application
platform	A character string indicating the platform.
version	A character vector of versions to remove. Defaults to "ALL"

**Value**

Returns a logical vector indicating whether the removal of version was successful. Return is invisible.

**Examples**

```
## Not run:
appdir <- app_dir(appname, FALSE)
platforms <- LETTERS[1:4]
versions <- LETTERS[5:7]
makedirs <- file.path(appdir, outer(platforms, versions, file.path))
chk <- vapply(makedirs, dir.create, logical(1), recursive = TRUE)
appver <- list_versions(appname)
rm_version(appname, platforms[2], versions[1:2])
unlink(appdir, recursive = TRUE)

## End(Not run)
```

---

unzip.tar\_dlfiles      *Unzip/Untar downloaded files*

---

**Description**

Unzip/Untar downloaded files. Keeps the original zip file

**Usage**

```
unzip.tar_dlfiles(dlfiles, chmod = FALSE)
```

**Arguments**

dlfiles	A data.frame of files by platform and indicating whether they were processed
chmod	change the mode of the unarchived file/files to "755" so they are executable on unix like systems.

**Value**

Returns a list of character vectors indicating files processed

**Examples**

```
## Not run:
ymlfile <- system.file("exdata", "sampleapp.yml", package = "binman")
trdata <- system.file("testdata", "test_dlres.Rdata", package = "binman")
load(trdata)
testthat::with_mock(
  `htr::GET` = function(...) {
    test_llres
  },
  `base::dir.create` = function(...) {
    TRUE
  },
  `utils::unzip` = function(zipfile, ...) {
    zipfile
  },
  procyaml <- process_yaml(ymlfile)
)
procyaml

## End(Not run)
```

# Index

app\_dir, [2](#)  
assign\_directory, [3](#)  
  
binman, [3](#)  
  
download\_files, [4](#), [7-9](#)  
  
list\_versions, [5](#)  
  
noproc\_dlfiles, [5](#)  
  
predl\_bitbucket\_downloads, [6](#)  
predl\_github\_assets, [7](#)  
predl\_google\_storage, [8](#)  
process\_yaml, [10](#)  
  
rm\_platform, [11](#)  
rm\_version, [11](#)  
  
unzipitar\_dlfiles, [12](#)