Package: consolidatePacks (via r-universe)

June 9, 2026

Type Package

Title Eliminate '@import' by Incorporating Dependencies Directly into the Package

Version 1.0.0

Date 2026-06-03

Maintainer Barry Zeeberg <barryz2013@gmail.com>

Depends R (>= 4.2.0)

Imports vprint, stringr, utils, devtools

Description The purpose of this package is to remove the '@import' dependence of an external package by consolidating the functions into your package. This may be necessary when the '@import' package is decommissioned by CRAN, and you do not want your dependent package to also be decommissioned. The functions in this package recursively retrieve dependencies in the external package. It also performs the other needed bookkeeping, such as retrieving .Rd files in the man subdirectory.

License GPL (>= 2)

Encoding UTF-8

VignetteBuilder knitr

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Barry Zeeberg [aut, cre]

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

Date/Publication 2026-06-09 08:44:10 UTC

RemoteUrl https://github.com/cran/consolidatePacks

RemoteRef HEAD

RemoteSha 1f52b936af2ea59a668b6b7e28ccb4f654280695

Contents

`addCode` *addCode*

Description

add needed functions to package .R file

Usage

```
addCode(pack, flist, r, verbose = 2)
```

Arguments

`pack`	character string path name of the p package directory
`flist`	return value of getCodes()
`r`	character string name of the package containing the needed functions
`verbose`	integer parameter for vprint()

Value

returns no value but has side effect of saving a file to disk

Examples

```
## Not run:
# run this with your own packages
dir1<-"/Users/barryzeeberg/personal/hearts/"
dir2<-"hearts_card_game_bayesian_inference/packages/foodwebWrapper"
pack2Copy<-sprintf("%s/%s",dir1,dir2)
pack<-workingCopy(pack2Copy)
p<-"/Users/barryzeeberg/personal/mvbutil_download/mvbutils/"
flist<-getCodes(p,c("globalVariables","foodweb","%&%"))
addCode(pack,flist,"mvbutils")

## End(Not run)
```

---

addImports *addImports*

Description

copy imports from r to packCopy

Usage

```
addImports(r, packCopy, verbose = 5)
```

Arguments

r	character string path name package containing the dependencies
packCopy	character string path name of the p package
verbose	integer parameter for vprint()

Value

returns no value but has side effect of updating some files in packCopy directory

Examples

```
## Not run:
# run this with your own packages
r<-"/Users/barryzeeberg/personal/mvbutil_download/mvbutils/"
dir1<-"/Users/barryzeeberg/personal/hearts/"
dir2<-"hearts_card_game_bayesian_inference/packages/foodwebWrapper"
pack2Copy<-sprintf("%s/%s",dir1,dir2)
packCopy<-workingCopy(pack2Copy)
addImports(r,packCopy)

## End(Not run)
```

addImportsToDESCRIPTION *addImportsToDESCRIPTION*

Description

add imports to packCopy package .R file

Usage

```
addImportsToDESCRIPTION(pack, need, verbose = 5)
```

Arguments

pack	character string path name of the p package
need	return value of retrieveImports()
verbose	integer parameter for vprint()

Value

returns no value, but has side effect of adding imports to the DESCRIPTION file

Examples

```
## Not run:
# run this with your own packages
dir1<-"/Users/barryzeeberg/personal/hearts/"
dir2<-"hearts_card_game_bayesian_inference/packages/foodwebWrapper"
pack2Copy<-sprintf("%s/%s",dir1,dir2)
packCopy<-workingCopy(pack2Copy)
addImportsToDESCRIPTION(packCopy,need=c("xyz","abc"))

## End(Not run)
```

addImportsToR *addImportsToR*

Description

add imports to packCopy package .R file

Usage

```
addImportsToR(pack, need, verbose = 5)
```

Arguments

pack	character string path name of the p package
need	return value of retrieveImports()
verbose	integer parameter for vprint()

Value

returns no value, but has side effect of adding imports to the .R file

Examples

```
## Not run:
# run this with your own packages
dir1<-"/Users/barryzeeberg/personal/hearts/"
dir2<-"hearts_card_game_bayesian_inference/packages/foodwebWrapper"
pack2Copy<-sprintf("%s/%s",dir1,dir2)
packCopy<-workingCopy(pack2Copy)
addImportsToR(packCopy,need=c("xyz","abc"))

## End(Not run)
```

---

addMan *addMan*

Description

copy .Rd files between two package directories

Usage

```
addMan(man, packCopy, funcs, verbose = 2)
```

Arguments

man	character string path name of the directory supplying the man files
packCopy	character string path name of the p package directory
funcs	vector of character strings, names of functions whose .Rd are needed
verbose	integer parameter for vprint()

Value

returns no values, but has side effect of modifying some files in packCopy

Examples

```
## Not run:
# run this with your own packages
dir1<-"/Users/barryzeeberg/personal/hearts/"
dir2<-"hearts_card_game_bayesian_inference/packages/foodwebWrapper"
pack2Copy<-sprintf("%s/%s",dir1,dir2)
packCopy<-workingCopy(pack2Copy)
man<-"/Users/barryzeeberg/personal/mvbutil_download/mvbutils/"
funcs<-"pos"
addMan(man,packCopy,funcs)

## End(Not run)
```

`checkConsistency` *checkConsistency*

Description

check whether it is ok for user to proceed

Usage

```
checkConsistency(f, r, packCopy, verbose = 5)
```

Arguments

`f`	character string name of the function whose dependencies are to be resolved
`r`	character string path name package containing the dependencies
`packCopy`	character string name of the p package
`verbose`	integer parameter for vprint()

Value

returns no values, but has side effect of stopping execution if error is detected

Examples

```
## Not run:
# run this with your own packages
checkConsistency(f="foodwebWrapper",r="mvbutils",packCopy="foodwebWrapper")

## End(Not run)
```

codeR *codeR*

Description

convert contents of funcs into a vector of character strings containing code for functions

Usage

```
codeR(funcs)
```

Arguments

funcs	list of character vectors containing code for functions

Value

returns a vector of character strings containing code for functions

Examples

```
l<-list()
l[["fake"]]<-"line of fake"
x<-codeR(l)
```

dependsRec *dependsRec*

Description

recursively retrieve dependencies

Usage

```
dependsRec(f, functions, n, ofile, maxRec = 10, verbose = 2)
```

Arguments

f	character string name of a function
functions	vector of character strings names of functions
n	integer designating the level of the recursion
ofile	character string path name of an output file
maxRec	integer terminate recursion after this level
verbose	integer parameter for vprint()

Details

both() requires that the package in which f is defined should be loaded in the search path

Value

returns no values but has side effect of writing a file to disk

Examples

```
## Not run:
# best to run this via dependsRecDriver()

## End(Not run)
```

`dependsRecDriver` *dependsRecDriver*

Description

retrieve names of required functions

Usage

```
dependsRecDriver(f, p, verbose = 2)
```

Arguments

`f`	character string name of a function whose dependencies are to be resolved
`p`	character string name of package to be used for resolving dependencies in f
`verbose`	integer parameter for vprint()

Value

returns a vector of character strings containing the names of required functions

Examples

```
## Not run:
# run this with your own packages
# a couple levels of recursion
v<-dependsRecDriver("foodweb","mvbutils",2)
# a lot of levels of recursion
v<-dependsRecDriver("HTGM4Ddriver","HTGM4D",2)

## End(Not run)
```

drDriver *drDriver*

Description

invoke functions to incorporate dependencies into package

Usage

```
drDriver(f, r, pack2Copy, ck = TRUE, verbose = c(1, 2, 4, 5, 6))
```

Arguments

f	character string name of the function whose dependencies are to be resolved
r	character string path name of the package containing the dependencies
pack2Copy	character string path name of the package directory that contains function f
ck	Boolean if TRUE perform check() at the end of the processing stream
verbose	integer parameter for vprint()

Details

The purpose of this package is to remove the @import dependence of an external package by consolidating the functions into your package. This may be necessary when the @import package is decommissioned by CRAN, and you do not want your dependent package to also be decommissioned.

Because changes are made to files in the R package directory files, a copy is automatically made of the input parameter pack2Copy

Packages r and pack2Copy should both be loaded in the search() path using library() The contents of the .R file from the package r might need to be copied and pasted into the R Console window or the RStudio Console window, in order to pick up functions that are not exported.

Some @import lines from the r file might need to be manually added to the .R file of the packCopy package directory after running drDriver()

Value

returns no values, but has side effect of modifying some files in packCopy

Examples

```
## Not run:
# run this with your own packages

# or (for this example and others in this package)

# download previous archived version of foodwebWrapper
# package foodwebWrapper_1.1.0.tar.gz from
```

```
# https://cran.r-project.org/src/contrib/Archive/foodwebWrapper/
# and
# download previous archived version of mvbutils package
# mvbutils_2.8.232.tar.gz from
# https://cran.r-project.org/src/contrib/Archive/mvbutils/

f<-"foodwebWrapper"
r<-"/Users/barryzeeberg/personal/mvbutil_download/mvbutils/"
dir1<-"/Users/barryzeeberg/personal/hearts/"
dir2<-"hearts_card_game_bayesian_inference/packages/foodwebWrapper"
pack2Copy<-sprintf("%s/%s",dir1,dir2)
drDriver(f,r,pack2Copy,ck=FALSE,verbose=2)

## End(Not run)
```

---

editF *editF*

Description

minor adjustment of the retrieved code to be suitable for inclusion in package .R file

Usage

```
editF(fname, f)
```

Arguments

fname	character string containing function name
f	vector of character strings comprising function code

Value

returns modified version of function code

Examples

```
editF(fname="fake",f="fake line")
```

`getCode` *getCode*

Description

retrieve the code for a given function from the .R file

Usage

```
getCode(pack, f, verbose = c(5, 6))
```

Arguments

`pack`	character string path name of the p package directory
`f`	character string name of the function to be retrieved
`verbose`	integer param passed to vprint()

Value

returns the code for a given function

Examples

```
## Not run:
# run this with your own packages
code<-getCode("/Users/barryzeeberg/personal/mvbutil_download/mvbutils/","globalVariables")

## End(Not run)
```

`getCodes` *getCodes*

Description

retrieve the code for given functions from the .R file

Usage

```
getCodes(pack, fs, verbose = c(5, 6))
```

Arguments

`pack`	character string path name of the p package directory
`fs`	vector of character string names of the functions to be retrieved
`verbose`	integer param passed to vprint()

Details

```
# print(class(flist))
# [1] "list"
# print(length(flist))
# [1] 10
# print(names(flist))
 # [1] "foodweb"     "find.funs"    "%is.a%"      "extract.named" "%matching%"   "named"       "lsall"
 # [8] "%except%"      "mcachees"      "pos"
# print(class(flist[[1]]))
# [1] "<-"
# x<-deparse(flist[[1]])
# print(class(x))
# [1] "character"
# print(length(x))
# [1] 45
# print(x[1:10])
 # [1] "\"foodweb\" <- function(funs, where = 1, charlim = 80, prune = character(0), "
 # [2] "    rprune, ancestors = TRUE, descendents = TRUE, plotting = TRUE, "
 # [3] "    plotmath = FALSE, generics = c(\"c\", \"print\", \"plot\", \"[\"), "
 # [4] "   lwd = 0.5, xblank = 0.18, border = \"transparent\", boxcolor = \"white\", "
 # [5] "    textcolor = \"black\", color.lines = TRUE, highlight = \"red\", "
 # [6] "    ...) {"
 # [7] "    oldpar <- par(..., no.readonly = TRUE)"
# [8] "    on.exit(par(oldpar))"
# [9] "    charlim <- charlim/par(\"cex\")"
# [10] "    par(lwd = lwd)"
```

Value

returns a list of the codes for the given functions. The elements of the list are of unusual class "call" or "<-". See details for more info.

Examples

```
## Not run:
# run this with your own packages
pack<-"/Users/barryzeeberg/personal/mvbutil_download/mvbutils/"
codes<-getCodes(pack,c("globalVariables","foodweb","%&%"))

## End(Not run)
```

`removeColons` *removeColons*

Description

remove : or :: or ::: references to a specified package from .R file

Usage

```
removeColons(pack, remove)
```

Arguments

pack	character string path name of the p package directory
remove	character string name of package to be removed

Value

returns no values, but has side effect of modifying some files in packCopy

Examples

```
## Not run:
# run this with your own packages
dir1<-"/Users/barryzeeberg/personal/hearts/"
dir2<-"hearts_card_game_bayesian_inference/packages/foodwebWrapper"
pack2Copy<-sprintf("%s/%s",dir1,dir2)
pack<-workingCopy(pack2Copy)
remove<-"mvbutils"
removeColons(pack,remove)

## End(Not run)
```

---

removeImport *removeImport*

Description

remove #' @import from .R file and remove import from DESCRIPTION file

Usage

```
removeImport(pack, remove, verbose = c(2, 5))
```

Arguments

pack	character string path name of the p package directory
remove	character string name of package to be removed from @import
verbose	integer parameter for vprint()

Value

returns no values, but has side effect of modifying some files in packCopy

Examples

```
## Not run:
# run this with your own packages
dir1<-"/Users/barryzeeberg/personal/hearts/"
dir2<-"hearts_card_game_bayesian_inference/packages/foodwebWrapper"
pack2Copy<-sprintf("%s/%s",dir1,dir2)
pack<-workingCopy(pack2Copy)
remove<-"mvbutils"
removeImport(pack,remove,2)

## End(Not run)
```

`retrieveFuncs` *retrieveFuncs*

Description

retrieve the code in a set of functions from a loaded package

Usage

```
retrieveFuncs(funcs)
```

Arguments

`funcs`	vector of character strings containing the names of functions

Details

although this function works, it is better to use getCodes(). getCodes() parses the .R file, whereas retrieveFuncs() can process only the functions that are in the global environment, so it misses funcitons that are not exported.

Value

returns a list of vectors of character strings containing function code

Examples

```
## Not run:
# run this with your own packages
retrieveFuncs("foodweb")

## End(Not run)
```

retrieveImports *retrieveImports*

Description

retrieve imports from r package

Usage

```
retrieveImports(r, packCopy)
```

Arguments

r	character string name of package containing the dependencies
packCopy	character string name of the p package

Value

returns a vector of character strings containing the names of the needed import packages

Examples

```
## Not run:
# run this with your own packages
retrieveImports(r="mvbutils",packCopy="foodwebWrapper")

## End(Not run)
```

workingCopy *workingCopy*

Description

make a working copy of the orginal R package directory

Usage

```
workingCopy(pack)
```

Arguments

pack	character string path name of the original package directory

Value

returns the pathname of the working copy

Examples

```
## Not run:
# run this with your own packages
dir1<-"/Users/barryzeeberg/personal/hearts/"
dir2<-"hearts_card_game_bayesian_inference/packages/foodwebWrapper"
pack2Copy<-sprintf("%s/%s",dir1,dir2)
packCopy<-workingCopy(pack2Copy)
print(sprintf("Working copy of R package directory is %s",packCopy))

## End(Not run)
```

Index