# Package 'BiocPkgTools'

October 16, 2019

**Type** Package

**Title** Collection of simple tools for learning about Bioc Packages

**Version** 1.2.0

**Date** 2019-04-29

**Description** Bioconductor has a rich ecosystem of metadata around
packages, usage, and build status. This package is a simple
collection of functions to access that metadata from R. The goal
is to expose metadata for data mining and value-added functionality
such as package searching, text mining, and analytics on
packages.

**Depends** htmlwidgets

**Imports** BiocManager, biocViews, tibble, stringr, rvest, rex, dplyr,
xml2, readr, httr, htmltools, DT, tools, utils, igraph, tidyr,
jsonlite, gh, RBGL, graph, magrittr

**VignetteBuilder** knitr

**Suggests** BiocStyle, knitr, rmarkdown, testthat, tm, SnowballC,
pdftools, visNetwork, clipr

**License** MIT + file LICENSE

**BugReports** https://github.com/seandavi/BiocPkgTools/issues/new

**URL** https://github.com/seandavi/BiocPkgTools

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.1.1

**biocViews** Software, Infrastructure

**git_url** https://git.bioconductor.org/packages/BiocPkgTools

**git_branch** RELEASE_3_9

**git_last_commit** d32d3dc

**git_last_commit_date** 2019-05-02

**Date/Publication** 2019-10-15

**Author** Shian Su [aut, ctb],
Lori Shepherd [ctb],
Marcel Ramos [ctb],
Martin Morgan [ctb],

1

Vince Carey [ctb],
Sean Davis [aut, cre]

**Maintainer** Sean Davis <seandavi@gmail.com>

# R topics documented:

---

biocBuildEmail *Create and copy e-mail package notification template to clipboard*

---

### Description

The `biocBuildEmail` function provides a template for notifying maintainers of errors in the Bioconductor Build System (BBS). This convenience function returns the body of the email from a template within the package and provides a copy in the clipboard.

### Usage

```
biocBuildEmail(pkg, version = BiocManager::version(),
  emailTemplate = .getTemplatePath())
```

### Arguments

| | |
|---|---|
| pkg | The name of the package in trouble |
| version | character(1) the version number as used to access the online build report. For example, "3.6". The default is the "current version" as specified in `BiocManager::version`. Note that this is a character() variable, not a number. |
| emailTemplate | The path to the email template. |

**Value**

a character string of the email

---

| biocBuildReport | *Tidy Bioconductor build report results* |
|---|---|

---

**Description**

The online Bioconoductor build reports are great for humans to look at, but they are not easily computable. This function scrapes HTML and text files available from the build report online pages to generate a tidy data frame version of the build report.

**Usage**

```
biocBuildReport(version = as.character(BiocManager::version()))
```

**Arguments**

version        character(1) the version number as used to access the online build report. For example, "3.6". The default is the "current version" as specified in `BiocManager::version`. Note that this is a character() variable, not a number.

**Value**

a `tbl_df` object with columns pkg, version, author, commit, date, node, stage, and result.

**Examples**

```
# Set the stage--what version of Bioc am
# I using?
BiocManager::version()

latest_build = biocBuildReport()
head(latest_build)
```

---

| biocDownloadStats | *get bioconductor download stats* |
|---|---|

---

**Description**

get bioconductor download stats

**Usage**

```
biocDownloadStats()
```

**Details**

Note that bioconductor package download stats are not version-specific.

**Value**

a data.frame of download stats for all bioconductor packages, in tidy format

**Examples**

```
biocDownloadStats()
```

---

biocExplore                   *Explore Bioconductor packages interactively*

---

**Description**

Explore Bioconductor packages through an interactive bubble plot. Click on bubbles to bring up additional information about the package. Size and proximity to center of a bubble is based on the downloads the package has in the past month.

**Usage**

```
biocExplore(top = 500, ...)
```

**Arguments**

top           maximum number of packages displayed in any biocView

...           parameters passed to htmlwidgets::createWidget()

**Value**

bubble plot of Bioconductor packages

---

biocPkgList                   *Get full bioc software package listing, with details*

---

**Description**

The BiocViews-generated VIEWS file is available for bioconductor release and devel repositories. It contains quite a bit more information from the package DESCRIPTION files than the PACKAGES file. In particular, it contains biocViews annotations and URLs for vignettes and developer URLs.

**Usage**

```
biocPkgList(version = BiocManager::version(), repo = "BioCsoft",
  addBiocViewParents = TRUE)
```

## Arguments

| | |
|---|---|
| version | The requested bioconductor version. Will default to use the BiocManager defaults (ie., `version()`). |
| repo | The requested biooconductor repository. The default will be the Bioconductor software repository: BioCsoft. Available repos include: "BioCsoft", "BioCann", "BioCexp", "BioCworkflows", and "CRAN". Note that not all repos are available for all versions, particularly older versions (but who would use those, right?). |
| addBiocViewParents | |
| | logical(), whether to add all biocViews parents to biocViews annotations. |

## Details

Since packages are annotated with the most specific views, the default functionality here is to add parent terms for all views for each package. For example, in the bioCsoft repository, all packages will have at least "Software" added to their biocViews. If one wants to stick to only the most specific terms, set `addBiocViewParents` to `FALSE`.

## Value

an object of class `tbl_df`.

## Examples

```
bpkgl = biocPkgList()
bpkgl
unlist(bpkgl[1,'Depends'])

# Get a list of all packages that
# import "GEOquery"
library(dplyr)
bpkgl %>%
  filter(Package=='GEOquery') %>%
  pull(c('importsMe'))
```

---

| BiocPkgTools | *BiocPkgTools: Examine and analyze Bioconductor package metadata* |
|---|---|

---

## Description

Bioconductor has a rich ecosystem of metadata around packages, usage, and build status. This package is a simple collection of functions to access that metadata from R. The goal is to expose metadata for data mining and value-added functionality such as package searching, text mining, and analytics on packages.

## For developers

The [biocBuildReport](#) function returns a computable form of the Bioconductor Build Report.

## For users

The `biocDownloadStats` function gets Bioconductor download stats, allowing users to quickly find commonly used packages. The `biocPkgList` is useful for getting a complete listing of all Bioconductor packages.

## Infrastructure

Bioconductor packages all have Digital Object Identifiers (DOIs). This package contains basic infrastructure for creating, updating, and de-referencing DOIs.

---

buildPkgDependencyDataFrame

*Work with Bioconductor package dependencies*

---

## Description

Bioconductor is built using an extensive set of core capabilities and data structures. This leads to package developers depending on other packages for interoperability and functionality. This function extracts package dependency information from `biocPkgList` and returns a tidy `data.frame` that can be used for analysis and to build graph structures of package dependencies.

## Usage

```
buildPkgDependencyDataFrame(dependencies = c("Depends", "Imports",
  "Suggests"), ...)
```

## Arguments

dependencies     character() vector including one or more of "Depends", "Imports", or "Suggests". Default is to include all possibilities.

...              parameters passed along to `biocPkgList`

## Value

a `data.frame` (also a `tbl_df`) of S3 class "biocDepDF" including columns "Package", "dependency", and "edgetype".

## Note

This function requires network access.

## See Also

See `buildPkgDependencyIgraph`, `biocPkgList`.

## Examples

```
# performs a network call, so must be online.
library(BiocPkgTools)
depdf = buildPkgDependencyDataFrame()
head(depdf)
library(dplyr)
# filter to include only "Imports" type
# dependencies
imports_only = depdf %>% filter(edgetype=='Imports')

# top ten most imported packages
imports_only %>% select(dependency) %>%
  group_by(dependency) %>% tally() %>%
  arrange(desc(n))

# Bioconductor packages doing the largest
# amount of importing
largest_importers = imports_only %>%
  select(Package) %>%
  group_by(Package) %>% tally() %>%
  arrange(desc(n))

# not sure what these packages do. Join
# to their descriptions
biocPkgList() %>% select(Package, Description) %>%
  left_join(largest_importers) %>% arrange(desc(n)) %>%
  head()
```

---

```
buildPkgDependencyIgraph
```
                    *Work with package dependencies as a graph*

---

## Description

Package dependencies represent a directed graph (though Bioconductor dependencies are not an acyclic graph). This function simply returns an igraph graph from the package dependency data frame from a call to buildPkgDependencyDataFrame or any tidy data frame with rows of (Package, dependency) pairs. Additional columns are added as igraph edge attributes (see graph_from_data_frame).

## Usage

```
buildPkgDependencyIgraph(pkgDepDF)
```

## Arguments

pkgDepDF          a tidy data frame. See description for details.

## Value

an igraph directed graph. See the igraph package for details of what can be done.

## See Also

See buildPkgDependencyDataFrame, graph_from_data_frame, inducedSubgraphByPkgs, subgraphByDegree,
igraph-es-indexing, igraph-vs-indexing

## Examples

```
library(igraph)

pkg_dep_df = buildPkgDependencyDataFrame()

# at this point, filter or join to manipulate
# dependency data frame as you see fit.

g = buildPkgDependencyIgraph(pkg_dep_df)
g

# Look at nodes and edges
head(V(g)) # vertices
head(E(g)) # edges

# subset graph by attributes


head(sort(degree(g, mode='in'), decreasing=TRUE))
head(sort(degree(g, mode='out'), decreasing=TRUE))
```

---

dataciteXMLGenerate          *Bioc datacite xml generator*

---

## Description

Bioc datacite xml generator

## Usage

```
dataciteXMLGenerate(pkg)
```

## Arguments

pkg                name of bioc package

## Value

an xml element

---

firstInBioc *When did a package enter Bioconductor?*

---

### Description

This function uses the biocDownloadStats data to \*approximate\* when a package entered Bioconductor. Note that the download stats go back only to 2009.

### Usage

```
firstInBioc(download_stats)
```

### Arguments

download_stats  a data.frame from `biocDownloadStats`

---

generateBiocPkgDOI *Generate a DOI for a bioconductor package*

---

### Description

This function makes calls out to the EZID API (v2) described here: `https://ezid.lib.purdue.edu/doc/apidoc.2.html`. The function creates a new DOI for a bioc package (cannot already exist). The target URL for the DOI is the short Bioconductor package URL.

### Usage

```
generateBiocPkgDOI(pkg, authors, pubyear, testing = TRUE)
```

### Arguments

pkg       character(1) package name

authors   character vector of authors (will be "pasted" together)

pubyear   integer(1) publication year

testing   logical(1) If true, will use the apitest user with the password apitest. These DOIs will expire. The same apitest:apitest combination can be used to login to the EZID website for doing things using the web interface. If false, the Bioconductor-specific user credentials should be in the correct environment variables

### Details

The login information for the "real" Bioconductor account should be stored in the environment variables "EZID_USERNAME" and "EZID_PASSWORD".

The GUI is available here: `https://doi.datacite.org/`.

### Value

The DOI as a character(1) vector.

### Examples

```
## Not run:
  x = generateBiocPkgDOI('RANDOM_TEST_PACKAGE','Sean Davis',1972)

## End(Not run)
```

---

getBiocVignette *Download a Bioconductor vignette*

---

### Description

The actual vignette path is available using `biocPkgList`.

### Usage

```
getBiocVignette(vignettePath, destfile = tempfile(),
  version = BiocManager::version())
```

### Arguments

| | |
|---|---|
| vignettePath | character(1) the additional path information to get to the vignette |
| destfile | character(1) the file location to store the vignette |
| version | chacter(1) such as "3.7", defaults to user version |

### Value

character(1) the filename of the downloaded vignette

### Examples

```
x = biocPkgList()
tmp = getBiocVignette(x$vignettes[[1]][1])
tmp

## Not run:
library(pdftools)
y = pdf_text(tmp)
y = paste(y,collapse=" ")
library(tm)
v = VCorpus(VectorSource(y))
library(magrittr)

v <- v %>%
    tm_map(stripWhitespace) %>%
    tm_map(content_transformer(tolower)) %>%
    tm_map(removeWords, stopwords("english")) %>%
    tm_map(stemDocument)
dtm = DocumentTermMatrix(v)
inspect(DocumentTermMatrix(v,
    list(dictionary = as.character(x$Package))))

## End(Not run)
```

---

getPackageInfo     *Generate needed information to create DOI from a package directory.*

---

### Description

Generate needed information to create DOI from a package directory.

### Usage

```
getPackageInfo(dir)
```

### Arguments

dir     character(1) Path to package

### Value

a data.frame

---

get_bioc_data     *Get data from bioconductor*

---

### Description

Get data from bioconductor

### Usage

```
get_bioc_data()
```

### Value

json string containing bioconductor package details

### Examples

```
bioc_data <- get_bioc_data()
```

githubDetails *Get package details from GitHub*

### Description

For packages that live on github, we can mine further details. This function returns the github details for the listed packages.

### Usage

```
githubDetails(pkgs, sleep = 0)
```

### Arguments

pkgs          a character() vector of username/repo for one or more github repos, such as 'seandavi/GEOquery'.

sleep         numeric() denoting the number of seconds to sleep between GitHub API calls. Since GitHub rate limits its APIs, it might be necessary to either use small chunks of packages iteratively or to supply a non-zero argument here. See the 'details' section for a better solution using GitHub tokens.

### Details

The gh function is used to do the fetching. If the number of packages supplied to this function is large (>40 or so), it is possible to run into problems with API rate limits. The gh package uses the environment variable "GITHUB_PAT" (for personal access token) to authenticate and then provide higher rate limits. If you run into problems with rate limits, set sleep to some small positive number to slow queries. Alternatively, create a Personal Access Token on GitHub and register it. See the gh package for details.

### Examples

```
pkglist = biocPkgList()

# example of "pkgs" format.
head(pkglist$URL)

gh_list = githubURLParts(pkglist$URL)
gh_list = gh_list[!is.null(gh_list$user_repo),]

head(gh_list$user_repo)

ghd = githubDetails(gh_list$user_repo[1:5])
lapply(ghd, '[[', "stargazers")
```

---

githubURLParts          *Extract github user and repo name from github URL*

---

### Description

Extract github user and repo name from github URL

### Usage

```
githubURLParts(urls)
```

### Arguments

urls                a `character()` vector of urls.

### Value

a `data.frame` with four columns:

- urlThe original github URL

- user_repoThe github "username/repo", combined

- userThe github username

- repoThe github repo name

### Examples

```
# find github URL details for
# Bioc packages
bpkgl = biocPkgList()
urldetails = githubURLParts(bpkgl$URL)
urldetails = urldetails[!is.na(urldetails$url),]
head(urldetails)
```

---

inducedSubgraphByPkgs   *Return a minimal subgraph based on package name(s)*

---

### Description

Find the subgraph induced by including specific packages. The induced subgraph is the graph that includes the named packages and all edges connecting them. This is useful for a developer, for example, to examine her packages and their intervening dependencies.

### Usage

```
inducedSubgraphByPkgs(g, pkgs, pkg_color = "red")
```

## Arguments

| | |
|---|---|
| g | an igraph graph, typically created by [buildPkgDependencyIgraph](buildPkgDependencyIgraph) |
| pkgs | character() vector of packages to include. Package names not included in the graph are ignored. |
| pkg_color | character(1) giving color of named packages. Other packages in the graph that fall in connecting paths will be colored as the igraph default. |

## Examples

```
library(igraph)
g = buildPkgDependencyIgraph(buildPkgDependencyDataFrame())
g2 = inducedSubgraphByPkgs(g, pkgs=c('GenomicFeatures',
'TCGAbiolinksGUI', 'BiocGenerics', 'org.Hs.eg.db', 'minfi', 'limma'))
g2
V(g2)

plot(g2)
```

---

problemPage                    *generate hyperlinked HTML for build reports for Bioc packages*

---

## Description

This is a quick way to get an HTML report of a developer's packages. The function is keyed to filter based on maintainer name.

## Usage

```
problemPage(authorPattern = "V.*Carey", ver = "devel",
  includeOK = FALSE)
```

## Arguments

| | |
|---|---|
| authorPattern | character(1) regexp used with grep() to filter author field of package DESCRIPTION for listing |
| ver | character(1) version tag for Bioconductor |
| includeOK | logical(1) include entries from the build report that are listed as "OK". Default FALSE will result in only those entries that are in WARNING or ERROR state. |

## Value

DT::datatable call; if assigned to a variable, must evaluate to get the page to appear

## Author(s)

Vince Carey

## Examples

```
if (interactive()) problemPage()
```

---

subgraphByDegree *Subset graph by degree*

---

### Description

While the inducedSubgraphByPkgs returns the subgraph with the minimal connections between named packages, this function takes a vector of package names, a degree (1 or more) and returns the subgraph(s) that are within degree of the package named.

### Usage

```
subgraphByDegree(g, pkg, degree = 1, ...)
```

### Arguments

| | |
|---|---|
| g | an igraph graph, typically created by buildPkgDependencyIgraph |
| pkg | character(1) package name from which to measure degree. |
| degree | integer(1) degree, limit search for adjacent vertices to this degree. |
| ... | passed on to distances |

### Value

an igraph graph, with only nodes and their edges within degree of the named package

### Examples

```
g = buildPkgDependencyIgraph(buildPkgDependencyDataFrame())
g2 = subgraphByDegree(g, 'GEOquery')
g2
```

# Index