# Package 'BiocPkgTools'

April 12, 2022

**Type** Package

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

**Version** 1.12.2

**Date** 2021-05-22

**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** BiocFileCache, BiocManager, biocViews, tibble, magrittr,
methods, rlang, tidyselect, stringr, rvest, dplyr, xml2, readr,
httr, htmltools, DT, tools, utils, igraph, tidyr, jsonlite, gh,
RBGL, graph

**VignetteBuilder** knitr

**Suggests** BiocStyle, knitr, rmarkdown, testthat, tm, SnowballC,
visNetwork, clipr, blastula, kableExtra, DiagrammeR,
SummarizedExperiment

**License** MIT + file LICENSE

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

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

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.1.2

**SystemRequirements** mailsend-go

**biocViews** Software, Infrastructure

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

**git_branch** RELEASE_3_14

**git_last_commit** 19280c6

**Author** Shian Su [aut, ctb],
     Lori Shepherd [ctb],
     Marcel Ramos [ctb],
     Felix G.M. Ernst [ctb],
     Charlotte Soneson [ctb],
     Martin Morgan [ctb],
     Vince Carey [ctb],
     Sean Davis [aut, cre]

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

# R **topics documented:**

anacondaDownloadStats *Get download statistics for Bioconductor packages distributed via Anaconda.*

## Description

Get download statistics for Bioconductor packages distributed via Anaconda.

## Usage

```
anacondaDownloadStats()
```

## Details

Anaconda provide daily download counts for all software packages they distribute. These are summarised into monthly tables of counts and made available from https://github.com/grimbough/anaconda-download-stats This function provides a mechanism to download these monthly counts for Bioconductor packages distributed through Anaconda.

## Value

A `data.frame` of download statistics for all Bioconductor packages distributed by Anaconda, in tidy format. Note: Anaconda do not provide counts for unique IP addresses. This column is listed as NA for all packages to provide continuity with data from Bioconductor.org obtained by `biocDownloadStats`. The counts are updated monthly, so do not expect to see counts for the current month.

## Author(s)

Mike L. Smith

## Examples

```
anacondaDownloadStats()
```

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 = c("release", "devel"),
  PS = character(1L),
  dry.run = TRUE,
  emailTemplate = templatePath(),
  core.name = NULL,
  core.email = NULL,
  core.id = NULL,
  textOnly = FALSE,
  resend = FALSE,
  verbose = FALSE,
  credFile = "~/.blastula_creds"
)

sentHistory()
```

## Arguments

| | |
|---|---|
| pkg | character(1) The name of the package in trouble |
| version | character() A vector indicating which version of Bioconductor the package is failing in (either 'release' or 'devel'; defaults to both) |
| PS | character(1) Postscript, an additional note to the recipient of the email (i.e., the package maintainer) |
| dry.run | logical(1) Display the email without sending to the recipient. It only works for HTML email reports and ignored when 'textOnly=TRUE' |
| emailTemplate | character(1) The path to the email template Rmd file as obtained by 'templatePath()'. A custom template can be provided as file path. |
| core.name | character(1) The full name of the core team member |
| core.email | character(1) The Roswell Park email of the core team member |
| core.id | character(1) The internal identifier for the Roswell employee. This ID usually matches '^[A-Z]2[0-9]5' for more recent identifiers. |
| textOnly | logical(1) Whether to return the text of the email only. This avoids the use of the 'blastula' package and adds the text to the system clipboard if the 'clipr' package is installed (default: FALSE) |
| resend | logical(1) Whether to force a resend of the email |
| verbose | logical(1) Whether to output full email information from 'smtp_send' (when 'dry.run' is 'FALSE' and 'blastula' is installed) |
| credFile | character(1) An optional file generated by the 'blastula::create_smtp_creds_file' function containing email auth info (default: "~/.blastula_creds"). |

## Value

A character string of the email

**sentHistory**

Check the history of emails sent

---

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 = BiocManager::version())
```

### Arguments

version          character(1) the character version number as used to access the online build report. For example, "3.14". The default is the "current version" as given by BiocManager::version(). Note that this is a character vector of length one and 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 statistics*

---

### Description

Get Bioconductor download statistics

### Usage

```
biocDownloadStats(
  pkgType = c("all", "software", "data-experiment", "workflows", "data-annotation")
)
```

### Arguments

pkgType          character(1) Either one of 'software', 'data-experiment', 'workflows', or 'data-annotation' (defaults to 'all' or 'software')

### Details

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

### Value

A `data.frame` of download statistics 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 = 500L, ...)
```

## Arguments

| | |
|---|---|
| `top` | maximum number of packages displayed in any biocView |
| `...` | parameters passed to `htmlwidgets::createWidget()` |

## Value

A bubble plot of Bioconductor packages

---

| | |
|---|---|
| `biocMaintained` | *Bioconductor Maintained Packages* |

---

## Description

List all the packages associated with a maintainer. By default, it will return all packages associated with the `maintainer@bioconductor.org` email.

## Usage

```
biocMaintained(
  version = BiocManager::version(),
  main = "maintainer@bioconductor\\.org"
)
```

## Arguments

| | |
|---|---|
| `version` | character(1) the character version number as used to access the online build report. For example, "3.14". The default is the "current version" as given by `BiocManager::version()`. Note that this is a character vector of length one and not a number. |
| `main` | character(1) The regex for searching through the Maintainer column as obtained from `biocPkgList()`. |

## Examples

```
biocMaintained()
```

---

biocPkgList                    *Get full Bioconductor 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'))
```

---

| biocPkgRanges | *Grab build report results from BUILD_STATUS_DB for a particular* |
| --- | --- |
| | *package range* |

---

### Description

Grab build report results from BUILD_STATUS_DB for a particular package range

### Usage

```
biocPkgRanges(
  start,
  end,
  condition = c("ERROR", "WARNINGS"),
  phase = "buildsrc",
  version = c("devel", "release")
)
```

### Arguments

| | |
| --- | --- |
| start | character(1) alphabetically first package name in range |
| end | character(1) alphabetically last package name in range |
| condition | character(1) condition string, typically 'ERROR' or 'WARNING' |
| phase | character(1) string for phase of event: 'install', 'checksrc', or 'buildsrc' (default) |
| version | character(1) string indication Bioconductor version, either 'devel' (default) or 'release' |

### Author(s)

Vincent J. Carey

### Examples

```
## Not run:
biocPkgRanges(
    start = "a4", end = "CMA",
    condition = "ERROR", version = "devel"
)

## End(Not run)
```

---

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.

---

BiocPkgTools-cache          *Manage cache for BiocPkgTools*

---

### Description

Managing user data is important to allow use of email functions such as 'biocBuildEmail' and made
easy with 'BiocFileCache'.

### Usage

```
setCache(
  directory = tools::R_user_dir("BiocPkgTools", "cache"),
  verbose = TRUE,
  ask = interactive()
)

pkgToolsCache(...)
```

## Arguments

| directory | The file location where the cache is located. Once set future downloads will go to this folder. |
|---|---|
| verbose | Whether to print descriptive messages |
| ask | logical (default TRUE when interactive session) Confirm the file location of the cache directory |
| ... | For `pkgToolsCache`, arguments are passed to `setCache` |

## pkgToolsCache

Get the directory location of the cache. It will prompt the user to create a cache if not already created. A specific directory can be used via `setCache`.

## setCache

Specify the directory location of the data cache. By default, it will got to the user's home/.cache/R and "appname" directory as specified by `tools::R_user_dir` (with package="BiocPkgTools" and which="cache").

---

| biocRevDepEmail | *Notify downstream maintainers of changes in upstream package* |
|---|---|

---

## Description

The 'biocRevDepEmail' function collects all the emails of the reverse dependencies and sends a notification that an upstream package has been deprecated or removed. It uses a template found in 'inst/resources' with the 'templatePath()' function.

## Usage

```
biocRevDepEmail(
  pkg,
  PS = character(1L),
  dry.run = TRUE,
  emailTemplate = templatePath("revdepnote"),
  core.name = NULL,
  core.email = NULL,
  core.id = NULL,
  textOnly = FALSE,
  verbose = FALSE,
  credFile = "~/.blastula_creds"
)
```

## Arguments

| | |
|---|---|
| pkg | character(1) The name of the package for whose reverse dependencies are to be checked and notified. |
| PS | character(1) Postscript, an additional note to the recipient of the email (i.e., the package maintainer) |
| dry.run | logical(1) Display the email without sending to the recipient. It only works for HTML email reports and ignored when 'textOnly=TRUE' |
| emailTemplate | character(1) The path to the email template Rmd file as obtained by 'templatePath()'. A custom template can be provided as file path. |
| core.name | character(1) The full name of the core team member |
| core.email | character(1) The Roswell Park email of the core team member |
| core.id | character(1) The internal identifier for the Roswell employee. This ID usually matches '^[A-Z]2[0-9]5' for more recent identifiers. |
| textOnly | logical(1) Whether to return the text of the email only. This avoids the use of the 'blastula' package and adds the text to the system clipboard if the 'clipr' package is installed (default: FALSE) |
| verbose | logical(1) Whether to output full email information from 'smtp_send' (when 'dry.run' is 'FALSE' and 'blastula' is installed) |
| credFile | character(1) An optional file generated by the 'blastula::create_smtp_creds_file' function containing email auth info (default: "~/.blastula_creds"). |

## Examples

```
biocRevDepEmail("FindMyFriends", dry.run = TRUE, textOnly = TRUE)
```

---

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 "Sug-
               gests". 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", "depen-
dency", 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))
```

| class-dependencies | *Retrieve Class relationships* |
|---|---|

### Description

As the title says it should do something with class relationships

### Usage

```
buildClassDepGraph(class, includeUnions = FALSE)

buildClassDepData(class, includeUnions = FALSE)

buildClassDepFromPackage(pkg, includeUnions = FALSE)

plotClassDep(class, includeUnions = FALSE)

plotClassDepData(data)

plotClassDepGraph(g)
```

### Arguments

| | |
|---|---|
| class | a single character value defining a 'S4' class name |
| includeUnions | TRUE or FALSE: Should union definitions included in the result? (default: FALSE) |
| pkg | a single character value defining a package name |
| data | a data.frame with compatible columns. See output of buildClassDepData |
| g | an igraph object with compatible edge attributes. See output of buildClassDepGraph |

### Examples

```
library("SummarizedExperiment")
depData <- buildClassDepData("RangedSummarizedExperiment")
depData
g <- buildClassDepGraph("RangedSummarizedExperiment")
plotClassDepGraph(g)
```

| CRANstatus | *Check the CRAN build report page and email a notification* |
|---|---|

### Description

The CRANstatus function allows users to check the status of a package and send an email report of any failures.

## Usage

```
CRANstatus(
  pkg,
  core.name = NULL,
  core.email = NULL,
  core.id = NULL,
  to.mail = "maintainer@bioconductor.org",
  dry.run = TRUE,
  emailTemplate = templatePath("cranreport")
)
```

## Arguments

| | |
|---|---|
| pkg | character(1) The name of the package in trouble |
| core.name | character(1) The full name of the core team member |
| core.email | character(1) The Roswell Park email of the core team member |
| core.id | character(1) The internal identifier for the Roswell employee. This ID usually matches '^[A-Z]2[0-9]5' for more recent identifiers. |
| to.mail | The email of the CRAN report recipient |
| dry.run | logical(1) Display the email without sending to the recipient. It only works for HTML email reports and ignored when 'textOnly=TRUE' |
| emailTemplate | character(1) The path to the email template Rmd file as obtained by 'templatePath()'. A custom template can be provided as file path. |

---

dataciteXMLGenerate          *Bioc datacite XML generator*

---

## Description

Bioc datacite XML generator

## Usage

```
dataciteXMLGenerate(pkg)
```

## Arguments

| | |
|---|---|
| pkg | The name of a Bioconductor 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](#)

### Examples

```
dls <- biocDownloadStats()
tail(firstInBioc(dls))
```

---

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](https://ezid.lib.purdue.edu/doc/apidoc.2.html). The function creates a new DOI for a Bioconductor 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

A 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
# Bioconductor 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` |
| 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)
```

---

pkgCombDependencyGain   *Calculate dependency gain achieved by excluding combinations of packages*

---

### Description

Calculate dependency gain achieved by excluding combinations of packages

### Usage

```
pkgCombDependencyGain(pkg, depdf, maxNbr = 3L)
```

## Arguments

| | |
|---|---|
| pkg | character, the name of the package for which we want to estimate the dependency gain |
| depdf | a tidy data frame with package dependency information obtained through the function buildPkgDependencyDataFrame |
| maxNbr | numeric, the maximal number of direct dependencies to leave out simultaneously |

## Value

A data frame with three columns: ExclPackages (the excluded direct dependencies), NbrExcl (the number of excluded direct dependencies), DepGain (the dependency gain from excluding these direct dependencies)

## Author(s)

Charlotte Soneson

## Examples

```
depdf <- buildPkgDependencyDataFrame(
  dependencies=c("Depends", "Imports"),
  repo=c("BioCsoft", "CRAN")
)
pcd <- pkgCombDependencyGain('GEOquery', depdf, maxNbr = 3L)
head(pcd[order(pcd$DepGain, decreasing = TRUE), ])
```

---

pkgDepImports *Report package imported functionality*

---

## Description

Function adapted from 'itdepends::dep_usage_pkg' at https://github.com/r-lib/itdepends to obtain the functionality imported and used by a given package.

## Usage

```
pkgDepImports(pkg)
```

## Arguments

| | |
|---|---|
| pkg | character() name of the package for which we want to obtain the functionality calls imported from its dependencies and used within the package. |

## Details

Certain imported elements, such as built-in constants, will not be identified as imported functionality by this function.

## Value

A tidy data frame with two columns:

- pkg: name of the package dependency.
- fun: name of the functionality call imported from the the dependency in the column pkg and used within the analyzed package.

## Author(s)

Robert Castelo

## Examples

```
pkgDepImports('BiocPkgTools')
```

---

pkgDepMetrics                  *Report package dependency burden*

---

## Description

Elaborate a report on the dependency burden of a given package.

## Usage

```
pkgDepMetrics(pkg, depdf)
```

## Arguments

| | |
|---|---|
| pkg | character() name of the package for which we want to obtain metrics on its dependency burden. |
| depdf | a tidy data frame with package dependency information obtained through the function buildPkgDependencyDataFrame. |

## Value

A tidy data frame with different metrics on the package dependency burden. More concretely, the following columns:

- ImportedAndUsed: number of functionality calls imported and used in the package.
- Exported: number of functionality calls exported by the dependency.

- Usage: (ImportedAndUsedx 100) / Exported. This value provides an estimate of what fraction of the functionality of the dependency is actually used in the given package.
- DepOverlap: Similarity between the dependency graph structure of the given package and the one of the dependency in the corresponding row, estimated as the [Jaccard index](#) between the two sets of vertices of the corresponding graphs. Its values goes between 0 and 1, where 0 indicates that no dependency is shared, while 1 indicates that the given package and the corresponding dependency depend on an identical subset of packages.
- DepGainIfExcluded: The 'dependency gain' (decrease in the total number of dependencies) that would be obtained if this package was excluded from the list of direct dependencies.

The reported information is ordered by the Usage column to facilitate the identification of dependencies for which the analyzed package is using a small fraction of their functionality and therefore, it could be easier remove them. To aid in that decision, the column DepOverlap reports the overlap of the dependency graph of each dependency with the one of the analyzed package. Here a value above, e.g., 0.5, could, albeit not necessarily, imply that removing that dependency could substantially lighten the dependency burden of the analyzed package.

An NA value in the ImportedAndUsed column indicates that the function pkgDepMetrics() could not identify what functionality calls in the analyzed package are made to the dependency.

## Author(s)

Robert Castelo

Charlotte Soneson

## Examples

```
depdf <- buildPkgDependencyDataFrame(
  dependencies=c("Depends", "Imports"),
  repo=c("BioCsoft", "CRAN")
)
pkgDepMetrics('BiocPkgTools', depdf)
```

---

pkgDownloadStats *Get Bioconductor download statistics for a package*

---

## Description

Get Bioconductor download statistics for a package

## Usage

```
pkgDownloadStats(
  pkg,
  pkgType = c("software", "data-experiment", "workflows", "data-annotation"),
  years = format(Sys.time(), "%Y")
)
```

## Arguments

| | |
|---|---|
| pkg | character(1) The name of a Bioconductor package |
| pkgType | character(1) Either one of 'software', 'data-experiment', 'workflows', or 'data-annotation' (defaults to 'all' or 'software') |
| years | numeric(), character() A vector of years from which to obtain download statistics (defaults to current year) |

## Value

A `tibble` of download statistics

## Examples

```
pkgDownloadStats("GenomicRanges")
```

---

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

---

## Description

This is a quick way to get an HTML report of packages maintained by a specific developer or which depend directly on a specified package. The function is keyed to filter based on either the maintainer name or by using the 'Depends', 'Suggests' and 'Imports' fields in package descriptions.

## Usage

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

## Arguments

| | |
|---|---|
| authorPattern | character(1) regexp used with grep() to filter author field of package DESCRIPTION for listing |
| dependsOn | character(1) name of a Bioconductor package. The function will return the status of packages that directly depend on this package Can only be used when 'authorPattern' is the empty string. |
| 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, Mike L. Smith

## Examples

```
if (interactive()) {
  problemPage()
  problemPage(dependsOn = "limma")
}
```

---

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
```

---

templatePath                    *Obtain the location of available email templates*

---

## Description

These templates are used with 'biocBuildEmail' to notify maintainers regarding package errors and final deprecation warning.

## Usage

```
templatePath(type = c("buildemail", "deprecation", "cranreport", "revdepnote"))
```

## Arguments

type            character(1) Either one of 'buildemail', 'cranreport', 'deprecation', or 'revdep-note'

# Index