Package ‘packageRank’

November 10, 2023

Type Package

Title Computation and Visualization of Package Download Counts and Percentiles

Version 0.8.3

Date 2023-11-10

Maintainer Peter Li <lindbrook@gmail.com>

Description Compute and visualize the cross-sectional and longitudinal number and rank percentile of package downloads from Posit/RStudio's CRAN mirror.

URL https://github.com/lindbrook/packageRank

BugReports https://github.com/lindbrook/packageRank/issues

Depends R (>= 3.5)

License GPL (>= 2)

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.2.3

Imports cranlogs, curl, data.table (>= 1.12.2), ggplot2, ISOcodes, memoise, pkgsearch, RCurl, R.utils, rversions, sugrrants, tools

Suggests knitr, rmarkdown

NeedsCompilation no

Author Peter Li [aut, cre]

Repository CRAN

Date/Publication 2023-11-10 00:50:02 UTC

R topics documented:

`annualDownloads` *Count Total CRAN Download.*

Description

From RStudio's CRAN Mirror http://cran-logs.rstudio.com/

Usage

```
annualDownloads(start.yr = 2013, end.yr = 2022)
```

Arguments

`start.yr`	Numeric or Integer.
`end.yr`	Numeric or Integer.

`archivePackages` *Packages in CRAN archive.*

Description

Scrape https://cran.r-project.org/src/contrib/Archive/.

Usage

```
archivePackages(include.date = FALSE, multi.core = TRUE,
  dev.mode = FALSE)
```

Arguments

`include.date`	Logical. Return data frame with package name and last publication date.
`multi.core`	Logical or Numeric. `TRUE` uses `parallel::detectCores()`. `FALSE` uses one, single core. You can also specify the number logical cores.
`dev.mode`	Logical. Development mode uses parallel::parLapply().

`bioconductorDownloads` *Annual/monthly package downloads from Bioconductor.*

Description

Annual/monthly package downloads from Bioconductor.

Usage

```
bioconductorDownloads(packages = NULL, from = NULL, to = NULL,
  when = NULL, unit.observation = "month")
```

Arguments

`packages`	Character. Vector of package names.
`from`	Start date as `yyyy-mm` or `yyyy`.
`to`	End date as `yyyy-mm` or `yyyy`.
`when`	`"last-year"`, or `"year-to-date"` or `"ytd"`.
`unit.observation`	"year" or "month".

Examples

```
## Not run:
# all packages
bioconductorDownloads()

# entire history
bioconductorDownloads(packages = "clusterProfiler")

# year-to-date
bioconductorDownloads(packages = "clusterProfiler", when = "ytd")
bioconductorDownloads(packages = "clusterProfiler", when = "year-to-date")

# last 12 months
bioconductorDownloads(packages = "clusterProfiler", when = "last-year")

# from 2015 to current year
bioconductorDownloads(packages = "clusterProfiler", from = 2015)

# 2010 through 2015 (yearly)
bioconductorDownloads(packages = "clusterProfiler", from = 2010, to = 2015,
  unit.observation = "year")

# selected year (yearly)
bioconductorDownloads(packages = "clusterProfiler", from = 2015, to = 2015)

# selected year (monthly)
bioconductorDownloads(packages = "clusterProfiler", from = "2015-01", to = "2015-12")

# June 2014 through March 2015
bioconductorDownloads(packages = "clusterProfiler", from = "2014-06", to = "2015-03")

## End(Not run)
```

---

bioconductorRank *Package download counts and rank percentiles.*

Description

From bioconductor

Usage

```
bioconductorRank(packages = "monocle", date = "2019-01",
  count = "download")
```

Arguments

packages	Character. Vector of package name(s).
date	Character. Date. yyyy-mm
count	Character. "ip" or "download".

Value

An R data frame.

Examples

```
## Not run:
bioconductorRank(packages = "cicero", date = "2019-09")

## End(Not run)
```

---

blog.data *Blog post data.*

Description

archive.pkg_ver

archive.pkg_ver.filtered

cran.pkg_ver

cran.pkg_ver.filtered

dl.ct

dl.ct2

pkg.ct

pkg.ct2

oct.data

cholera.data

ggplot2.data

VR.data

smpl

smpl.histories

smpl.archive

smpl.archive.histories

ccode.ct

crosstab_2019_10_01

percentiles

top.n.oct2019

top.n.jul2020

download.country

october.downloads

july.downloads

```
cran.pkgs.oct
arch.pkgs.oct
cran.pkgs.jul
arch.pkgs.jul
pkg.history
```

Usage

```
blog.data
```

Format

A list with 29 elements.

`countryDistribution` *Tabulate package downloads by country.*

Description

From RStudio's CRAN Mirror http://cran-logs.rstudio.com/

Usage

```
countryDistribution(date = NULL, all.filters = FALSE, ip.filter = FALSE,
  triplet.filter = FALSE, small.filter = FALSE, sequence.filter = FALSE,
  size.filter = FALSE, memoization = TRUE, multi.core = TRUE,
  dev.mode = FALSE)
```

Arguments

`date`	Character. Date. "yyyy-mm-dd". NULL uses latest available log.
`all.filters`	Logical. Master switch for filters.
`ip.filter`	Logical.
`triplet.filter`	Logical.
`small.filter`	Logical. TRUE filters out downloads less than 1000 bytes.
`sequence.filter`	Logical.
`size.filter`	Logical.
`memoization`	Logical. Use memoization when downloading logs.
`multi.core`	Logical or Numeric. `TRUE` uses `parallel::detectCores()`. `FALSE` uses one, single core. You can also specify the number logical cores. Mac and Unix only.
`dev.mode`	Logical. Development mode uses parallel::parLapply().

Value

An R data frame.

countryPackage *Tabulate a country's package downloads.*

Description

From RStudio's CRAN Mirror http://cran-logs.rstudio.com/

Usage

```
countryPackage(country = "HK", date = NULL, all.filters = FALSE,
  ip.filter = FALSE, triplet.filter = FALSE, small.filter = FALSE,
  sequence.filter = FALSE, size.filter = FALSE, sort = TRUE,
  memoization = TRUE, multi.core = TRUE, dev.mode = FALSE)
```

Arguments

`country`	Character. country abbreviation.
`date`	Character. Date. "yyyy-mm-dd". NULL uses latest available log.
`all.filters`	Logical. Master switch for filters.
`ip.filter`	Logical.
`triplet.filter`	Logical.
`small.filter`	Logical.
`sequence.filter`	Logical. Set to FALSE.
`size.filter`	Logical. Set to FALSE.
`sort`	Logical. Sort by download count.
`memoization`	Logical. Use memoization when downloading logs.
`multi.core`	Logical or Numeric. `TRUE` uses `parallel::detectCores()`. `FALSE` uses one, single core. You can also specify the number logical cores. Mac and Unix only.
`dev.mode`	Logical. Development mode uses parallel::parLapply().

Note

"US" outlier 10 min with all filters!

`countsRanks` *Counts v. Rank Percentiles for 'cholera' for First Week of March 2020.*

Description

Document code for blog graph.

Usage

```
countsRanks(package = "cholera", size.filter = FALSE)
```

Arguments

`package`	Character.
`size.filter`	Logical.

`cranDownloads` *Daily package downloads from the RStudio CRAN mirror.*

Description

Enhanced implementation of cranlogs::cran_downloads().

Usage

```
cranDownloads(packages = NULL, when = NULL, from = NULL, to = NULL,
  check.package = TRUE, dev.mode = FALSE, fix.cranlogs = TRUE)
```

Arguments

`packages`	A character vector, the packages to query, or `NULL` for a sum of downloads for all packages. Alternatively, it can also be `"R"`, to query downloads of R itself. `"R"` cannot be mixed with packages.
`when`	`last-day`, `last-week` or `last-month`. If this is given, then `from` and `to` are ignored.
`from`	Start date as `yyyy-mm-dd`, `yyyy-mm` or `yyyy`.
`to`	End date as `yyyy-mm-dd`, `yyyy-mm` or `yyyy`.
`check.package`	Logical. Validate and "spell check" package.
`dev.mode`	Logical. Use validatePackage0() to scrape CRAN.
`fix.cranlogs`	Logical. Use RStudio logs to fix 8 dates with duplicated data in 'cranlogs' results.

Examples

```
## Not run:
cranDownloads(packages = "HistData")
cranDownloads(packages = "HistData", when = "last-week")
cranDownloads(packages = "HistData", when = "last-month")

# January 7 - 31, 2019
cranDownloads(packages = "HistData", from = "2019-01-07", to = "2019-01-31")

# February through March 2019
cranDownloads(packages = "HistData", from = "2019-02", to = "2019-03")

# 2020 year-to-date
cranDownloads(packages = "HistData", from = 2020)

## End(Not run)
```

`cranInflationPlot` *CRAN inflation plot.*

Description

Document code.

Usage

```
cranInflationPlot(dataset = "october")
```

Arguments

`dataset`	Character. "october" or "july" for October 2019 or July 2020.

`cranMirrors` *Scrape CRAN Mirrors data.*

Description

https://cran.r-project.org/mirrors.html

Usage

```
cranMirrors(description = FALSE)
```

Arguments

`description`	Logical. Mirror details.

cranPackages *Scrape CRAN package information.*

Description

Current version, date and size (source and binary).

Usage

```
cranPackages(binary = FALSE, bytes = FALSE, multi.core = TRUE)
```

Arguments

`binary`	Logical. Compute size of binary files.
`bytes`	Logical. Compute approximate numeric file size in bytes.
`multi.core`	Logical or Numeric. `TRUE` uses `parallel::detectCores()`. `FALSE` uses one, single core. You can also specify the number logical cores. Mac and Unix only.

Value

An R data frame.

currentTime *Compute Current Time in Selected Time Zone.*

Description

Compute Current Time in Selected Time Zone.

Usage

```
currentTime(tz = "Australia/Sydney")
```

Arguments

`tz`	Character. Local time zone. See OlsonNames() or use Sys.timezone().

`downloadsCountry` *Compute Downloads by Country Code.*

Description

Compute Downloads by Country Code.

Usage

```
downloadsCountry(month_cran_log, multi.core = TRUE)
```

Arguments

`month_cran_log`	Object.
`multi.core`	Logical or Numeric. `TRUE` uses `parallel::detectCores()`. `FALSE` uses one, single core. You can also specify the number logical cores to use. Note that due to performance considerations, the number of cores defaults to one on Windows.

`extractArchiveDate` *Extract a package's archive date.*

Description

Date a package is moved to Archive (if available).

Usage

```
extractArchiveDate(package)
```

Arguments

`package`	Character. Package name.

Value

An R data frame.

`fetchCranLog` *Fetch CRAN Logs.*

Description

Fetch CRAN Logs.

Usage

```
fetchCranLog(date, memoization = FALSE, dev.mode = FALSE)
```

Arguments

`date`	Character. Date. yyyy-mm-dd.
`memoization`	Logical. Use memoization when downloading logs.
`dev.mode`	Logical. Use Base R code.

`fetchRLog` *Fetch R download Logs.*

Description

Fetch R download Logs.

Usage

```
fetchRLog(date)
```

Arguments

`date`	Character. Date. yyyy-mm-dd.

filteredDownloads — *Filtered package downloads from the RStudio CRAN mirror (prototype).*

Description

ip, triplet, small, sequence and size filters.

Usage

```
filteredDownloads(packages = "HistData", date = NULL, all.filters = TRUE,
  ip.filter = FALSE, triplet.filter = FALSE, small.filter = FALSE,
  sequence.filter = FALSE, size.filter = FALSE, check.package = TRUE,
  memoization = TRUE, multi.core = TRUE)
```

Arguments

`packages`	Character. Vector of package name(s).
`date`	Character. Date. "yyyy-mm-dd". NULL uses latest available log.
`all.filters`	Logical. Master switch for filters.
`ip.filter`	Logical.
`triplet.filter`	Logical.
`small.filter`	Logical. TRUE filters out downloads less than 1000 bytes.
`sequence.filter`	Logical.
`size.filter`	Logical.
`check.package`	Logical. Validate and "spell check" package.
`memoization`	Logical. Use memoization when downloading logs.
`multi.core`	Logical or Numeric. `TRUE` uses `parallel::detectCores()`. `FALSE` uses one, single core. You can also specify the number logical cores. Mac and Unix only.

inflationPlot — *Inflation plots of effects of "small" downloads and prior versions for October 2019: 'cholera', 'ggplot2', and 'VR'.*

Description

Document code for blog graph.

Usage

```
inflationPlot(package = "cholera", filter = "size",
  legend.loc = "topleft")
```

Arguments

`package`	Character.
`filter`	Character. Size, version, or size and version
`legend.loc`	Character. Location of legend.

`inflationPlot2` *Inflation plots of effects of "small" downloads on aggregate CRAN downloads for October 2019 and July 2020.*

Description

Document code.

Usage

```
inflationPlot2(dataset = "october", filter = "small", wed = FALSE,
  subtitle = TRUE, legend.loc = "topleft")
```

Arguments

`dataset`	Character. "october" or "july" for October 2019 or July 2020.
`filter`	Character. "small", "ip", or "ip.small".
`wed`	Logical.
`subtitle`	Logical.
`legend.loc`	Character. Location of legend.

`ipCount` *Count number of IP addresses.*

Description

From RStudio's CRAN Mirror http://cran-logs.rstudio.com/

Usage

```
ipCount(date = NULL, memoization = TRUE, sort = TRUE)
```

Arguments

`date`	Character. Date. "yyyy-mm-dd". NULL uses latest available log.
`memoization`	Logical. Use memoization when downloading logs.
`sort`	Logical. Sort by download count.

`ipDownloads` *Unique package download counts by IP address.*

Description

From RStudio's CRAN Mirror http://cran-logs.rstudio.com/

Usage

```
ipDownloads(date = NULL, memoization = TRUE)
```

Arguments

`date`	Character. Date. "yyyy-mm-dd". NULL uses latest available log.
`memoization`	Logical. Use memoization when downloading logs.

`ipFilter` *Filter Out A-Z Campaigns from IPs with many unique package downloads.*

Description

Uses run length encoding, rle(), and k-means clustering, stats::kmeans().

Usage

```
ipFilter(cran_log, campaigns = TRUE, rle.depth = 100,
  case.sensitive = FALSE, multi.core = TRUE, dev.mode = dev.mode)
```

Arguments

`cran_log`	Object. Package log entries.
`campaigns`	Logical. Filter A-Z campaigns when checking IPs with high unique package download counts.
`rle.depth`	s Numeric. Ceiling for number of rows of run length encoding. Fewer rows means longer runs.
`case.sensitive`	Logical.
`multi.core`	Logical or Numeric. `TRUE` uses `parallel::detectCores()`. `FALSE` uses one, single core. You can also specify the number logical cores. Mac and Unix only.
`dev.mode`	Logical. Development mode uses parallel::parLapply().

`ipPackage` *Tabulate an IP's package downloads.*

Description

From RStudio's CRAN Mirror http://cran-logs.rstudio.com/

Usage

```
ipPackage(ip = 10, date = NULL, all.filters = FALSE, ip.filter = FALSE,
  triplet.filter = FALSE, small.filter = FALSE, sequence.filter = FALSE,
  size.filter = FALSE, sort = TRUE, memoization = TRUE,
  multi.core = TRUE, dev.mode = FALSE)
```

Arguments

`ip`	Numeric. ip_id.
`date`	Character. Date. "yyyy-mm-dd". NULL uses latest available log.
`all.filters`	Logical. Master switch for filters.
`ip.filter`	Logical.
`triplet.filter`	Logical.
`small.filter`	Logical. TRUE filters out downloads less than 1000 bytes.
`sequence.filter`	Logical.
`size.filter`	Logical.
`sort`	Logical. Sort by download count.
`memoization`	Logical. Use memoization when downloading logs.
`multi.core`	Logical or Numeric. `TRUE` uses `parallel::detectCores()`. `FALSE` uses one, single core. You can also specify the number logical cores. Mac and Unix only.
`dev.mode`	Logical. Development mode uses parallel::parLapply().

Note

ip = 10 is a tw top-level domain on 2020-07-09.

`localTime` *Compute Local Time from Coordinated Universal Time (UTC/GMT).*

Description

Compute Local Time from Coordinated Universal Time (UTC/GMT).

Usage

```
localTime(date = "2021-1-1", time = "12:00", tz = Sys.timezone())
```

Arguments

`date`	Character. Date "yyyy-mm-dd".
`time`	Character. Local time "hh:mm" or "hh:mm:ss".
`tz`	Character. Local time zone. See OlsonNames() or use Sys.timezone().

`logDate` *Compute Effective CRAN Log Date Based on Local and UTC Time (prototype).*

Description

RStudio CRAN Mirror Logs for previous day are posted at 17:00:00 UTC.

Usage

```
logDate(date = NULL, check.url = TRUE, tz = Sys.timezone(),
  upload.time = "17:00", warning.msg = TRUE, fix.date = TRUE)
```

Arguments

`date`	Character. Date of desired log `"yyyy-mm-dd"`. NULL returns date of latest available log.
`check.url`	Logical.
`tz`	Character. Time zone. See OlsonNames().
`upload.time`	Character. UTC upload time for logs "hh:mm" or "hh:mm:ss".
`warning.msg`	Logical. TRUE uses warning() if the function returns the date of the previous available log.
`fix.date`	Logical. Fix date when directly accessing RStudio logs.

Value

An R date object.

`logInfo` *Compute Availability, Date, Time of "Today's" Log.*

Description

Also checks availability of Posit/RStudio logs and 'cranlogs' data.

Usage

```
logInfo(tz = Sys.timezone(), upload.time = "17:00",
  show.available = FALSE)
```

Arguments

`tz`	Character. Local time zone. See OlsonNames() or use Sys.timezone().
`upload.time`	Character. UTC upload time for logs "hh:mm" or "hh:mm:ss".
`show.available`	Logical. Check available logs and results.

`monthlyLog` *Get CRAN logs for selected month.*

Description

Compute list of log files, 'lst', for packageVersionPercent().

Usage

```
monthlyLog(yr.mo = "2020-07")
```

Arguments

`yr.mo`	Character. "yyyy-mm".

Note

This is computationally intensive; you're downloading 30 odd files that are each around 50 MB in size (and creating a ~1.5 GB file)! Parallelization not practical; multiple attempts to connect to website causes problems. Truncates in-progress/future dates to yesterday's date. Automatically takes care of leap days (e.g., monthlyLog("2020-02").

packageArchive *Scrape package data from Archive.*

Description

Scrape package data from Archive.

Usage

```
packageArchive(package = "cholera", check.package = TRUE, size = FALSE)
```

Arguments

`package`	Character. Package name.
`check.package`	Logical. Validate and "spell check" package.
`size`	Logical. Include size of source file.

Value

An R data frame or NULL.

Examples

```
## Not run:
packageArchive(package = "HistData")
packageArchive(package = "adjustedcranlogs")  # No archived versions.

## End(Not run)
```

packageCountry *Package download counts by country.*

Description

From RStudio's CRAN Mirror http://cran-logs.rstudio.com/

Usage

```
packageCountry(packages = "cholera", date = NULL, all.filters = FALSE,
  ip.filter = FALSE, triplet.filter = FALSE, small.filter = FALSE,
  sequence.filter = FALSE, size.filter = FALSE, sort = TRUE,
  na.rm = FALSE, memoization = TRUE, check.package = TRUE,
  multi.core = TRUE, dev.mode = FALSE)
```

Arguments

`packages`	Character. Vector of package name(s).
`date`	Character. Date. "yyyy-mm-dd". NULL uses latest available log.
`all.filters`	Logical. Master switch for filters.
`ip.filter`	Logical.
`triplet.filter`	Logical.
`small.filter`	Logical.
`sequence.filter`	Logical.
`size.filter`	Logical.
`sort`	Logical. Sort by download count.
`na.rm`	Logical. Remove NAs.
`memoization`	Logical. Use memoization when downloading logs.
`check.package`	Logical. Validate and "spell check" package.
`multi.core`	Logical or Numeric. `TRUE` uses `parallel::detectCores()`. `FALSE` uses one, single core. You can also specify the number logical cores. Mac and Unix only.
`dev.mode`	Logical. Development mode uses parallel::parLapply().

`packageCRAN` *Scrape package data from CRAN.*

Description

Version, date and size (source file) of most recent publication.

Usage

```
packageCRAN(package = "cholera", check.package = TRUE, size = FALSE)
```

Arguments

`package`	Character. Package name.
`check.package`	Logical. Validate and "spell check" package.
`size`	Logical. Include size of source file.

Value

An R data frame or NULL.

Examples

```
## Not run:
packageCRAN(package = "HistData")
packageCRAN(package = "VR") # No version on CRAN (archived)

## End(Not run)
```

packageDistribution *Package Download Distribution.*

Description

Package Download Distribution.

Usage

```
packageDistribution(package = "HistData", date = NULL,
  all.filters = FALSE, ip.filter = FALSE, small.filter = FALSE,
  memoization = TRUE, check.package = TRUE, multi.core = TRUE,
  dev.mode = FALSE, threshold = 1000L)
```

Arguments

`package`	Character. Vector of package name(s).
`date`	Character. Date. "yyyy-mm-dd". NULL uses latest available log.
`all.filters`	Logical. Master switch for filters.
`ip.filter`	Logical.
`small.filter`	Logical. TRUE filters out downloads less than 1000 bytes.
`memoization`	Logical. Use memoization when downloading logs.
`check.package`	Logical. Validate and "spell check" package.
`multi.core`	Logical or Numeric. `TRUE` uses `parallel::detectCores()`. `FALSE` uses one, single core. You can also specify the number logical cores. Mac and Unix only.
`dev.mode`	Logical. Development mode uses parallel::parLapply().
`threshold`	Numeric. Threshold for small.filter in Bytes.

packageHistory *Extract package or R version history.*

Description

Date and version of all publications.

Usage

```
packageHistory(package = "cholera", check.package = FALSE)
```

Arguments

`package`	Character. Vector of package names (including "R").
`check.package`	Logical. Validate and "spell check" package.

packageLog — *Get Package Download Logs.*

Description

From RStudio's CRAN Mirror http://cran-logs.rstudio.com/

Usage

```
packageLog(packages = "cholera", date = NULL, all.filters = FALSE,
  ip.filter = FALSE, triplet.filter = FALSE, small.filter = FALSE,
  sequence.filter = FALSE, size.filter = FALSE, memoization = TRUE,
  check.package = TRUE, multi.core = TRUE, dev.mode = FALSE)
```

Arguments

`packages`	Character. Vector of package name(s).
`date`	Character. Date. "yyyy-mm-dd". NULL uses latest available log.
`all.filters`	Logical. Master switch for filters.
`ip.filter`	Logical.
`triplet.filter`	Logical.
`small.filter`	Logical. TRUE filters out downloads less than 1000 bytes.
`sequence.filter`	Logical.
`size.filter`	Logical.
`memoization`	Logical. Use memoization when downloading logs.
`check.package`	Logical. Validate and "spell check" package.
`multi.core`	Logical or Numeric. `TRUE` uses `parallel::detectCores()`. `FALSE` uses one, single core. You can also specify the number logical cores. Mac and Unix only.
`dev.mode`	Logical. Development mode uses parallel::parLapply().

Value

An R data frame.

packageRank — *Package download counts and rank percentiles (prototype).*

Description

From RStudio's CRAN Mirror http://cran-logs.rstudio.com/

Usage

```
packageRank(packages = "HistData", date = NULL, all.filters = FALSE,
  ip.filter = FALSE, small.filter = FALSE, memoization = TRUE,
  check.package = TRUE, multi.core = TRUE, dev.mode = FALSE,
  threshold = 1000L)
```

Arguments

`packages`	Character. Vector of package name(s).
`date`	Character. Date. "yyyy-mm-dd". NULL uses latest available log.
`all.filters`	Logical. Master switch for filters.
`ip.filter`	Logical.
`small.filter`	Logical. TRUE filters out downloads less than 1000 bytes.
`memoization`	Logical. Use memoization when downloading logs.
`check.package`	Logical. Validate and "spell check" package.
`multi.core`	Logical or Numeric. `TRUE` uses `parallel::detectCores()`. `FALSE` uses one, single core. You can also specify the number logical cores. Mac and Unix only.
`dev.mode`	Logical. Development mode uses parallel::parLapply().
`threshold`	Numeric. Threshold for small.filter in Bytes.

Value

An R data frame.

Examples

```
## Not run:
packageRank(packages = "HistData", date = "2020-01-01")
packageRank(packages = c("h2o", "Rcpp", "rstan"), date = "2020-01-01")

## End(Not run)
```

`packages_in_Archive` *Packages currently in Archive*

Description

Include inactive (retired) packages and previous versions of active packages.

Usage

```
packages_in_Archive(multi.core = TRUE)
```

Arguments

`multi.core`	Logical or Numeric. `TRUE` uses `parallel::detectCores()`. `FALSE` uses one, single core. You can also specify the number logical cores. See `vignette("Parallelization")` for details.

Value

An R vector.

`packages_observed_in_logs` *Packages observed in download logs.*

Description

Packages observed in download logs.

Usage

```
packages_observed_in_logs(date = NULL)
```

Arguments

`date`	Character. Date. "yyyy-mm-dd". NULL uses latest available log.

Value

An R vector.

`packages_on_CRAN` *Packages currently on CRAN.*

Description

Package name, version and date of publication.

Usage

```
packages_on_CRAN()
```

Value

An R data frame.

`packages_partitioned` *Partitioned CRAN and Archive Packages.*

Description

CRAN, Archive, Observed, CRAN & Archive, CRAN only and Archive only.

Usage

```
packages_partitioned(observed.downloads = FALSE, multi.core = TRUE)
```

Arguments

`observed.downloads`	Logical. Compute current observed package downloads.
`multi.core`	Logical or Numeric. `TRUE` uses `parallel::detectCores()`. `FALSE` uses one, single core. You can also specify the number logical cores. See `vignette("Parallelization")` for details.

Value

An R list.

`packageVersionPercent` *Compute data for versionPlot().*

Description

packageRank::blog.data or recompute random sample of packages.

Usage

```
packageVersionPercent(lst, yr.mo = "2020-07", multi.core = TRUE)
```

Arguments

`lst`	Object. List of CRAN download logs data frames. Use monthlyLog().
`yr.mo`	Character. "yyyy-mo". packageVersionsPercent(NULL, yr.mo)
`multi.core`	Logical or Numeric. `TRUE` uses `parallel::detectCores()`. `FALSE` uses one, single core. You can also specify the number logical cores. Mac and Unix only.

Examples

```
## Not run:
# To resample and recompute, set lst to NULL, specify a yr.mo:
packageVersionPercent(NULL, yr.mo = "2020-07")

Otherwise, you must provide a pre-computed lst of logs.

## End(Not run)
```

`plot.annualDownloads` *Plot method for annualDownloads().*

Description

Plot method for annualDownloads().

Usage

```
## S3 method for class 'annualDownloads'
plot(x, statistic = "count", pool.obs = FALSE,
  log.y = FALSE, sep.y = FALSE, nrow = 3, smooth = TRUE, span = 3/4,
  ...)
```

Arguments

`x`	object.
`statistic`	Character. "count" or "percent".
`pool.obs`	Logical.
`log.y`	Logical. Base 10 logarithm of y-axis.
`sep.y`	Logical. Separate, independent y-scales for each panel.
`nrow`	Numeric. Number of rows for ggplot2 facets.
`smooth`	Logical. Add smoother (loess).
`span`	Numeric. Smoothing parameter for geom_smooth(); c.f. stats::loess(span). 3/4 is built-in default.
`...`	Additional plotting parameters.

`plot.bioconductorDownloads` *Plot method for bioconductorDownloads().*

Description

Plot method for bioconductorDownloads().

Usage

```
## S3 method for class 'bioconductorDownloads'
plot(x, graphics = NULL,
  count = "download", cumulative = FALSE, points = "auto",
  smooth = FALSE, f = 2/3, span = 3/4, se = FALSE, log.y = FALSE,
  r.version = FALSE, same.xy = TRUE, multi.plot = FALSE,
  legend.loc = "topleft", ...)
```

Arguments

`x`	object.
`graphics`	Character. NULL, "base" or "ggplot2".
`count`	Character. "download" or "ip".
`cumulative`	Logical. Use cumulative counts.
`points`	Character of Logical. Plot points. "auto", TRUE, FALSE. "auto" for bioconductorDownloads(unit.observation = "month") with 24 or fewer months, points are plotted.
`smooth`	Logical. Add stats::lowess smoother.
`f`	Numeric. smoother window for stats::lowess(). For graphics = "base" only; c.f. stats::lowess(f)
`span`	Numeric. Smoothing parameter for geom_smooth(); c.f. stats::loess(span).

`se`	Logical. Works only with graphics = "ggplot2".
`log.y`	Logical. Logarithm of package downloads.
`r.version`	Logical. Add R release dates.
`same.xy`	Logical. Use same scale for multiple packages when graphics = "base".
`multi.plot`	Logical. Plot all data in a single window frame.
`legend.loc`	Character.
`...`	Additional plotting parameters.

Examples

```
## Not run:
plot(bioconductorDownloads())
plot(bioconductorDownloads(packages = "graph"))
plot(bioconductorDownloads(packages = "graph", from = 2010, to = 2015))
plot(bioconductorDownloads(packages = "graph", from = "2014-06", to = "2015-03"))
plot(bioconductorDownloads(packages = c("graph", "IRanges", "S4Vectors"), from = 2018))

## End(Not run)
```

`plot.bioconductorRank` *Plot method for bioconductorRank().*

Description

Plot method for bioconductorRank().

Usage

```
## S3 method for class 'bioconductorRank'
plot(x, graphics = NULL, log_count = TRUE, ...)
```

Arguments

`x`	An object of class "bioconductor_rank" created by `bioconductorRank()`.
`graphics`	Character. "base" or "ggplot2".
`log_count`	Logical. Logarithm of package downloads.
`...`	Additional plotting parameters.

Value

A base R or ggplot2 plot.

`plot.countryDistribution` *Plot top 10 package downloads by country domain.*

Description

Plot method for packageDistribution().

Usage

```
## S3 method for class 'countryDistribution'
plot(x, ...)
```

Arguments

`x`	An object of class "countryDistribution" created by `countryDistribution()`.
`...`	Additional plotting parameters.

`plot.countsRanks` *Plot method for countsRanks().*

Description

Plot method for countsRanks().

Usage

```
## S3 method for class 'countsRanks'
plot(x, ...)
```

Arguments

`x`	object.
`...`	Additional plotting parameters.

plot.cranDownloads *Plot method for cranDownloads().*

Description

Plot method for cranDownloads().

Usage

```
## S3 method for class 'cranDownloads'
plot(x, statistic = "count", graphics = "auto",
  points = "auto", log.y = FALSE, smooth = FALSE, se = FALSE,
  f = 1/3, span = 3/4, package.version = FALSE, r.version = FALSE,
  population.plot = FALSE, population.seed = as.numeric(Sys.Date()),
  multi.plot = FALSE, same.xy = TRUE, legend.location = "topleft",
  ip.legend.location = "topright", r.total = FALSE, dev.mode = FALSE,
  unit.observation = "day", multi.core = TRUE, ...)
```

Arguments

`x`	object.
`statistic`	Character. "count" or "cumulative".
`graphics`	Character. "auto", "base" or "ggplot2".
`points`	Character of Logical. Plot points. "auto", TRUE, FALSE.
`log.y`	Logical. Logarithm of package downloads.
`smooth`	Logical. Add smoother.
`se`	Logical. Works only with graphics = "ggplot2".
`f`	Numeric. smoother window for stats::lowess(). For graphics = "base" only; c.f. stats::lowess(f)
`span`	Numeric. Smoothing parameter for geom_smooth(); c.f. stats::loess(span).
`package.version`	Logical. Add latest package release dates.
`r.version`	Logical. Add R release dates.
`population.plot`	Logical. Plot population plot.
`population.seed`	Numeric. Seed for sample in population plot.
`multi.plot`	Logical.
`same.xy`	Logical. Use same scale for multiple packages when graphics = "base".
`legend.location`	Character.
`ip.legend.location`	Character. Location of in-progress legend.

`r.total`	Logical.
`dev.mode`	Logical. Use packageHistory0() to scrape CRAN.
`unit.observation`	Character. "year", "month", "week", or "day".
`multi.core`	Logical or Numeric. `TRUE` uses `parallel::detectCores()`. `FALSE` uses one, single core. You can also specify the number logical cores. Mac and Unix only.
`...`	Additional plotting parameters.

Value

A base R or ggplot2 plot.

Examples

```
## Not run:
plot(cranDownloads(packages = c("Rcpp", "rlang", "data.table")))
plot(cranDownloads(packages = c("Rcpp", "rlang", "data.table"), when = "last-month"))
plot(cranDownloads(packages = "R", from = "2020-01-01", to = "2020-01-01"))
plot(cranDownloads(packages = "R", from = 2020))

## End(Not run)
```

`plot.packageDistribution` *Plot method for packageDistribution().*

Description

Plot method for packageDistribution().

Usage

```
## S3 method for class 'packageDistribution'
plot(x, ...)
```

Arguments

`x`	An object of class "packageDistribution" created by `packageDistribution()`.
`...`	Additional plotting parameters.

`plot.packageRank` *Plot method for packageRank() and packageRank0().*

Description

Plot method for packageRank() and packageRank0().

Usage

```
## S3 method for class 'packageRank'
plot(x, graphics = NULL, log_count = TRUE, ...)
```

Arguments

`x`	An object of class "packageRank" created by `packageRank()`.
`graphics`	Character. "base" or "ggplot2".
`log_count`	Logical. Logarithm of package downloads.
`...`	Additional plotting parameters.

Value

A base R or ggplot2 plot.

Examples

```
## Not run:
plot(packageRank(packages = "HistData", date = "2020-01-01"))
plot(packageRank(packages = c("h2o", "Rcpp", "rstan"), date = "2020-01-01"))

## End(Not run)
```

`plot.packageVersionPercent` *Plot method for packageVersionPercent().*

Description

Plot method for packageVersionPercent().

Usage

```
## S3 method for class 'packageVersionPercent'
plot(x, ...)
```

Arguments

`x`	An object of class "packageVersions" created by `packageVersions()`.
`...`	Additional plotting parameters.

`plot.weeklyDownloads` *Plot method for weeklyDownloads().*

Description

Plot method for weeklyDownloads().

Usage

```
## S3 method for class 'weeklyDownloads'
plot(x, statistic = "percent",
  aggregation = "day", typical.value = "mean", nrow = 3L, ...)
```

Arguments

`x`	object.
`statistic`	Character. "count" or "percent".
`aggregation`	Character. "week" or "day".
`typical.value`	Character. "mean" or "median".
`nrow`	Numeric. Number of rows for ggplot2 facets.
`...`	Additional plotting parameters.

Examples

```
## Not run:
plot(weeklyDownloads())
plot(weeklyDownloads(n = 9), aggregation = "week")

## End(Not run)
```

`plotDownloadsCountry` *Plot Compute Downloads by Country Code.*

Description

Plot Compute Downloads by Country Code.

Usage

```
plotDownloadsCountry()
```

`plotTopCountryCodes` *Plot Top N Downloads by Country Code.*

Description

Plot Top N Downloads by Country Code.

Usage

```
plotTopCountryCodes(dataset = "october", second.place = FALSE)
```

Arguments

`dataset`	Character.
`second.place`	Logical. Annotate second place country.

`print.bioconductorDownloads` *Print method for bioconductorDownloads().*

Description

Print method for bioconductorDownloads().

Usage

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

Arguments

`x`	object.
`...`	Additional parameters.

`print.bioconductorRank` *Print method for bioconductorRank().*

Description

Print method for bioconductorRank().

Usage

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

Arguments

`x`	An object of class "bioconductor_rank" created by `bioconductorRank()`
`...`	Additional parameters.

`print.cranDownloads` *Print method for cranDownloads().*

Description

Print method for cranDownloads().

Usage

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

Arguments

`x`	object.
`...`	Additional parameters.

`print.packageDistribution`
Print method for packageDistribution().

Description

Print method for packageDistribution().

Usage

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

Arguments

`x`	An object of class "packageDistribution" created by `packageDistribution()`
`...`	Additional parameters.

`print.packageRank` *Print method for packageRank().*

Description

Print method for packageRank().

Usage

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

Arguments

`x`	An object of class "packageRank" created by `packageRank()`
`...`	Additional parameters.

`rstudio.logs`	*Eight RStudio Download Logs to Fix Duplicate Logs Errors in 'cran-logs'.*

Description

October 6-8, 2012; October 11, 2012; December 26-28; and January 1, 20113.

Usage

```
rstudio.logs
```

Format

```
date
time
size
r_version
r_arch
r_os
package
version
country
ip_id
```

`sequenceFilter`	*Filter downloads of full-sized sequential versions (prototype).*

Description

Filter downloads of full-sized sequential versions (prototype).

Usage

```
sequenceFilter(dat, packages, ymd, cores, download.time = 30,
  dev.mode = dev.mode)
```

Arguments

`dat`	Object.
`packages`	Object. An R vector of package names.
`ymd`	Date. Log date.
`cores`	Numeric. Number of cores to use.
`download.time`	Numeric. Package download time allowance (seconds).
`dev.mode`	Logical. Development mode uses parallel::parLapply().

`sizeFilter` *Filter out size anomalies (prototype).*

Description

Logs from RStudio's CRAN Mirror http://cran-logs.rstudio.com/

Usage

```
sizeFilter(dat, packages, cores, dev.mode = dev.mode)
```

Arguments

`dat`	Object. Package log entries.
`packages`	Character. Vector of package name(s).
`cores`	Integer. Number of cores for parallelization.
`dev.mode`	Logical. Development mode uses parallel::parLapply().

`smallFilter` *Filter out small downloads (prototype).*

Description

Filter out small downloads (prototype).

Usage

```
smallFilter(dat, threshold = 1000L, multi.core = TRUE,
  dev.mode = dev.mode)
```

Arguments

`dat`	Object. Package log entries.
`threshold`	Numeric. Bytes.
`multi.core`	Logical or Numeric. `TRUE` uses `parallel::detectCores()`. `FALSE` uses one, single core. You can also specify the number logical cores. Mac and Unix only.
`dev.mode`	Logical. Development mode uses parallel::parLapply().

summary.bioconductorDownloads

Summary method for bioconductorDownloads().

Description

Summary method for bioconductorDownloads().

Usage

```
## S3 method for class 'bioconductorDownloads'
summary(object, ...)
```

Arguments

`object`	Object.
`...`	Additional parameters.

summary.bioconductorRank

Summary method for bioconductorRank().

Description

Summary method for bioconductorRank().

Usage

```
## S3 method for class 'bioconductorRank'
summary(object, ...)
```

Arguments

`object`	Object. An object of class "bioconductor_rank" created by `bioconductorRank()`
`...`	Additional parameters.

Note

This is useful for directly accessing the data frame.

`summary.cranDownloads` *Summary method for cranDownloads().*

Description

Summary method for cranDownloads().

Usage

```
## S3 method for class 'cranDownloads'
summary(object, ...)
```

Arguments

`object`	Object.
`...`	Additional parameters.

Note

This is useful for directly accessing the data frame.

`summary.packageRank` *Summary method for packageRank().*

Description

Summary method for packageRank().

Usage

```
## S3 method for class 'packageRank'
summary(object, ...)
```

Arguments

`object`	Object. An object of class "packageRank" created by `packageRank()`
`...`	Additional parameters.

Note

This is useful for directly accessing the data frame.

topCountryCodes *Compute Top N Downloads by Country Code.*

Description

Compute Top N Downloads by Country Code.

Usage

```
topCountryCodes(month_cran_log, top.n = 5L, multi.core = TRUE)
```

Arguments

`month_cran_log`	Object.
`top.n`	Integer.
`multi.core`	Logical or Numeric. `TRUE` uses `parallel::detectCores()`. `FALSE` uses one, single core. You can also specify the number logical cores to use. Note that due to performance considerations, the number of cores defaults to one on Windows.

tripletFilter *Filter out small downloads triplets (prototype).*

Description

Logs from RStudio's CRAN Mirror http://cran-logs.rstudio.com/

Usage

```
tripletFilter(dat, time.window = 2, multi.core = TRUE,
  dev.mode = dev.mode)
```

Arguments

`dat`	Object. Package log entries.
`time.window`	Numeric. Seconds.
`multi.core`	Logical or Numeric. `TRUE` uses `parallel::detectCores()`. `FALSE` uses one, single core. You can also specify the number logical cores. Mac and Unix only.
`dev.mode`	Logical. Development mode uses parallel::parLapply().

`utc` *Compute Coordinated Universal Time (UTC/GMT) for Your Local Time.*

Description

Compute Coordinated Universal Time (UTC/GMT) for Your Local Time.

Usage

```
utc()
```

`utc0` *Compute Coordinated Universal Time (UTC/GMT) for Specified Local Time.*

Description

Compute Coordinated Universal Time (UTC/GMT) for Specified Local Time.

Usage

```
utc0(date = "2020-01-01", time = "12:00:00", tz = "Europe/Vienna")
```

Arguments

`date`	Character. Date "yyyy-mm-dd".
`time`	Character. Local time "hh:mm" or "hh:mm:ss".
`tz`	Character. Local time zone. See OlsonNames() or use Sys.timezone().

`versionPlot` *Version Plot.*

Description

Document code for blog graph.

Usage

```
versionPlot()
```

`weeklyDownloads` *Sample Weekly CRAN Downloads Data.*

Description

From RStudio's CRAN Mirror http://cran-logs.rstudio.com/

Usage

```
weeklyDownloads(start.yr = 2013, n = 50, multi.core = TRUE)
```

Arguments

`start.yr`	Numeric or Integer.
`n`	Numeric or Integer. Number of weeks (samples).
`multi.core`	Logical or Numeric. `TRUE` uses `parallel::detectCores()`. `FALSE` uses one, single core. You can also specify the number logical cores. Mac and Unix only.

Index