Package 'alm'

March 13, 2015

Title R Client for the Lagotto Altmetrics Platform

Description An R interface to the open source article level metrics platform <https://github.com/articlemetrics/lagotto/> created by the Public Library of Science (PLOS). A number of publishers are using the open source app created by PLOS, so you can drop in a different base URL to the functions in this package to get to not only PLOS data, but data for Crossref, and more as the open source PLOS software is used.

Version 0.4.0

Date 2015-03-13

License MIT + file LICENSE

URL https://github.com/ropensci/alm

BugReports http://www.github.com/ropensci/alm/issues

LazyData true

VignetteBuilder knitr

Imports ggplot2, plyr, stringr, reshape, reshape2, httr, grid, jsonlite, lubridate

Suggests testthat, roxygen2, rplos, knitr

Author Scott Chamberlain [aut, cre], Carl Boettiger [aut], Karthik Ram [aut], Fenner Martin [aut]

Maintainer Scott Chamberlain <myrmecocystus@gmail.com>

NeedsCompilation no

Repository CRAN

Date/Publication 2015-03-13 20:22:04

R topics documented:

`alm-package` *R client for the open source article level metrics Lagotto application.*

Description

R client for the open source article level metrics Lagotto application.

Details

An R interface to the RESTful API from the open source article level metrics software Lagotto, created by the Public Library of Science (PLOS). A number of publishers are using Lagotto, so you can drop in a different base URL to the functions in this package to get to not only PLOS data, but data for Crossref, and more.

Authenication was required, but has now been removed going foward in Lagotto. You only need API keys for a few of the data providers that are running old versions of Lagotto, and that will change as they upgrade their Lagotto sofware.

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

`alert_classes` *List the possible alert classes*

Description

List the possible alert classes

Usage

```
alert_classes()
```

alm-defunct *Defunct functions in alm*

Description

- alm_pubmedid: Function removed, you can get this info using alm_ids
- alm_pubmedcentid: Function removed, you can get this info using alm_ids
- almupdated: Function removed, you can get this info using alm_ids
- almdateupdated: Function removed, you can get this info using alm_ids
- alm_totals: Function removed, you can get this info using alm_ids

Get PubMed article ID by inputting the doi for the article.

Get PubMed Central article ID by inputting the doi for the article.

Date when alt-metrics for the article (by DOI) data was last updated.

Get the date when article was last updated.

Alt-metrics total citations from all sources.

Usage

```
alm_pubmedid(...)

alm_pubmedcentid(...)

almupdated(...)

almdateupdated(...)

alm_totals(...)
```

Arguments

...	Args

alm_alerts *Retrieve alerts data for article-level metrics (ALM).*

Description

Retrieve alerts data for article-level metrics (ALM).

Usage

```
alm_alerts(source_id = NULL, publisher_id = NULL, ids = NULL,
  class_name = NULL, level = NULL, q = NULL, unresolved = FALSE,
  per_page = 50, page = 1, user = NULL, pwd = NULL,
  api_url = "http://alm.plos.org/api/v4/alerts", ...)
```

Arguments

source_id	(character) Source to limtit alert search to.
publisher_id	(character) Metrics for articles by a given publisher, using the Crossref member_id.
ids	(character) Article identifiers
class_name	(character) Which error to get
level	(character) Alert level to limit search to. One of ERROR, WARN, INFO, or FATAL. Default: all.
q	(character) Query terms
unresolved	(logical) Whether to give back unresolved alerts only.
per_page	(integer) Number of records to return. Default: 50. Max: ?
page	(integer) Number from 1 to number of pages
user	(character) Username
pwd	(character) Password
api_url	URL to use.
...	Further args passed to httr::GET

Details

This function uses the alm.plos.org API by default. You can change which ALM app you use by specifying the url in the api_url parameter. It will likely be the same as the default http://alm.plos.org/api/v4/alerts, but just the alm.plos.org part will be different.

ALM installations on the most current version (3.4.7) can be used in this function, as of 2014-08-26 those are PLOS, ALM Labs, and Crossref Labs.

Examples

```
## Not run:
alm_alerts()
alm_alerts(q='mismatch')
alm_alerts(unresolved=TRUE)
alm_alerts(class_name='NoMethodError')
alm_alerts(class_name='ApiResponseTooSlowError')
alm_alerts(level='Error')
alm_alerts(ids="10.1371/journal.pone.0029797")

# curl debugging
library("httr")
alm_alerts(level='Error', config=verbose())
```

```
# paging
alm_alerts(per_page=2)

# by source_id
alm_alerts(source_id = "wos")

# by publisher_id
alm_alerts(publisher_id = 340)

## End(Not run)
```

`alm_datepub` *Get the date when the article was published.*

Description

Get the date when the article was published.

Usage

```
alm_datepub(doi = NULL, pmid = NULL, pmcid = NULL, wos = NULL,
  scp = NULL, url = NULL, get = NULL, key = NULL,
  api_url = "http://alm.plos.org/api/v5/articles", ...)
```

Arguments

`doi`	Digital object identifier for an article in PLoS Journals (character)
`pmid`	PubMed object identifier (numeric)
`pmcid`	PubMed Central object identifier (numeric)
`wos`	Web of Science identifier (character)
`scp`	Scopus identifier (character)
`url`	Canonical URL (character)
`get`	Get year, month, or day; if unspecified, whole date returned.
`key`	your PLoS API key, either enter, or loads from .Rprofile (character)
`api_url`	API endpoint, defaults to http://alm.plos.org/api/v3/articles (character)
`...`	optional additional curl options (debugging tools mostly)

Value

Date when article was published.

References

See a tutorial/vignette for alm at http://ropensci.org/tutorials/alm_tutorial.html

Examples

```
## Not run:
alm_datepub(doi='10.1371/journal.pone.0026871')
alm_datepub('10.1371/journal.pone.0026871', get='year')

# Provide more than one DOI
dois <- c('10.1371/journal.pone.0026871','10.1371/journal.pone.0048868',
'10.1371/journal.pone.0048705','10.1371/journal.pone.0048731')
alm_datepub(doi=dois, get="month")

## End(Not run)
```

`alm_events` *Retrieve PLoS article-level metrics (ALM) events.*

Description

Events are the details of the metrics that are counted related to PLoS papers.

Usage

```
alm_events(doi = NULL, pmid = NULL, pmcid = NULL, wos = NULL,
  scp = NULL, url = NULL, source_id = NULL, publisher_id = NULL,
  compact = TRUE, key = NULL,
  api_url = "http://alm.plos.org/api/v5/articles", ...)
```

Arguments

`doi`	Digital object identifier for an article in PLoS Journals (character)
`pmid`	PubMed object identifier (numeric)
`pmcid`	PubMed Central object identifier (numeric)
`wos`	Web of Science identifier (character)
`scp`	Scopus identifier (character)
`url`	Canonical URL (character)
`source_id`	(character) Name of source to get ALM information for. One source only. You can get multiple sources via a for loop or lapply-type call.
`publisher_id`	(character) Metrics for articles by a given publisher, using the Crossref `member_id`.
`compact`	(logical) Whether to make output compact or not. If TRUE (default), remove empty sources.
`key`	(character) Your API key, either enter, or loads from .Rprofile. Only required for PKP source, not the others.
`api_url`	API endpoint, defaults to http://alm.plos.org/api/v3/articles (character)
`...`	optional additional curl options (debugging tools mostly)

Details

You can only supply one of the parmeters doi, pmid, pmcid, and mendeley.

Query for as many articles at a time as you like. Though queries are broken up in to smaller bits of 30 identifiers at a time.

If you supply both the days and months parameters, days takes precedence, and months is ignored.

You can get events from many different sources. After calling alm_events, then index the output by the data provider you want. The options are: bloglines, citeulike, connotea, crossref, nature, postgenomic, pubmed, scopus, plos, researchblogging, biod, webofscience, pmc, facebook, mendeley, twitter, wikipedia, and scienceseeker.

Beware that some data source are not parsed yet, so there may be event data but it is not provided yet as it is so messy to parse.

See more info on PLOS's relative metrics event source here http://www.plosone.org/static/almInfo#relativeMetrics

Value

PLoS altmetrics as data.frame's.

References

See a tutorial/vignette for alm at http://ropensci.org/tutorials/alm_tutorial.html

Examples

```
## Not run:
# For one article
out <- alm_events(doi="10.1371/journal.pone.0029797")
names(out) # names of sources
# remove those with no data
out <- out[!out %in% c("sorry, no events content yet","parser not written yet")]
out["pmc"]] # get the results for PubMed Central
out["twitter"]] # get the results for twitter
out["plos_comments"]] # get the results for PLOS comments, sorta messy
out[c("twitter","crossref")] # get the results for two sources

# Another example
(out <- alm_events(doi="10.1371/journal.pone.0001543"))
# remove those with no data
out <- out[!out %in% c("sorry, no events content yet","parser not written yet")]
names(out)
out[['scopus']]
out[['mendeley']]
out[['figshare']]
out[['pubmed']]

# Two doi's
dois <- c('10.1371/journal.pone.0001543','10.1371/journal.pone.0040117')
out <- alm_events(doi=dois)
out[[1]]
```

```
out[[2]]
out[[1]][["figshare"]]$events

# Many pmcid's
out <- alm_events(pmcid=c(212692,2082661))
names(out)
out['212692']

# Many pmid's
out <- alm_events(pmid = c(19300479, 19390606, 19343216))
names(out)
out['19390606']

# Specify two specific sources
## You have to do so through lapply, or similar approach
lapply(c("crossref","twitter"),
   function(x) alm_events(doi="10.1371/journal.pone.0035869", source_id=x))

# Figshare data
alm_events(doi="10.1371/journal.pone.0069841", source_id='figshare')

# Datacite data
alm_events("10.1371/journal.pone.0012090", source_id='datacite')

# Reddit data
alm_events("10.1371/journal.pone.0015552", source_id='reddit')

# Wordpress data
alm_events("10.1371/journal.pcbi.1000361", source_id='wordpress')

# Articlecoverage data
alm_events(doi="10.1371/journal.pmed.0020124", source_id='articlecoverage')

# Articlecoveragecurated data
headfoo <- function(x) head(x$articlecoveragecurated$events)
headfoo(alm_events(doi="10.1371/journal.pone.0088278", source_id='articlecoveragecurated'))
headfoo(alm_events(doi="10.1371/journal.pmed.1001587", source_id='articlecoveragecurated'))

# F1000 Prime data
alm_events(doi="10.1371/journal.pbio.1001041", source_id='f1000')
dois <- c('10.1371/journal.pmed.0020124','10.1371/journal.pbio.1001041',
          '10.1371/journal.pbio.0040020')
res <- alm_events(doi = dois, source_id='f1000')
res[[3]]

# by source_id only
alm_events(source_id = "crossref")
alm_events(source_id = "reddit")

# by publisher_id only
alm_events(publisher_id = 340)

# search the software lagotto sever
```

```
urls <- c("https://github.com/najoshi/sickle","https://github.com/lh3/wgsim",
   "https://github.com/jstjohn/SeqPrep")
dat <- alm_events(url = urls, api_url = "http://software.lagotto.io/api/v5/articles")

## End(Not run)
```

`alm_ids` *Retrieve PLoS article-level metrics (ALM).*

Description

This is the main function to search the PLoS ALM (article-level metrics) API. See details for more information.

Usage

```
alm_ids(doi = NULL, pmid = NULL, pmcid = NULL, wos = NULL, scp = NULL,
  url = NULL, info = "totals", source_id = NULL, publisher_id = NULL,
  key = NULL, total_details = FALSE, sum_metrics = NULL, sleep = 0,
  api_url = "http://alm.plos.org/api/v5/articles", ...)
```

Arguments

`doi`	(character) Digital object identifier for an article in PLoS Journals
`pmid`	(numeric) PubMed object identifier
`pmcid`	(numeric) PubMed Central object identifier
`wos`	(character) Web of Science identifier
`scp`	(character) Scopus identifier
`url`	(character) Canonical URL. This is a URL for an object, not the URL for the Lagotto instance (see api_url)
`info`	One of totals, summary, or detail (default totals + sum_metrics data in a list). Not specifying anything (the default) returns data.frame of totals across data providers. (character). `info='detail'` returns all possible data, including slots for info, signposts, totals, and sum_metrics (see the sum_metrics parameter below). IMPORTANT: note, however, that you can only get by day metrics for articles published since May 2014.
`source_id`	(character) Name of source to get ALM information for. One source only. You can get multiple sources via a for loop or lapply-type call.
`publisher_id`	(character) Metrics for articles by a given publisher, using the Crossref `member_id`.
`key`	(character) Your API key, either enter, or loads from .Rprofile. Only required for PKP source, not the others.
`total_details`	(logical) If FALSE (the default) the standard totals data.frame is returned; if TRUE, the totals data is in a wide format with more details about the paper, including publication date, title, etc. If you set this to TRUE, the output should no longer with with `alm_plot`.

sum_metrics	(character) Just like the output you get from setting info='totals', you can get summary metrics by day (sum_metrics='day'), month (sum_metrics='month'), or year (sum_metrics='year'). IMPORTANT: note that you can only get by day metrics for articles published since May 2014.
sleep	Set a sleep time (in seconds). Only used for large calls where you may be in danger of upsetting the server gods, can you say 504 error?
api_url	(character) API endpoint, defaults to http://alm.plos.org/api/v5/articles
...	Curl options (debugging tools mostly) passed on to GET

Details

You can only supply one of the parmeters doi, pmid, pmcid, wos, scp, or url; and you must supply one of them. Query for as many articles at a time as you like. Though queries are broken up in to smaller bits of 50 identifiers at a time. If you supply days, months and/or year parameters, days takes precedence over months and year.

Value

PLoS altmetrics as data.frame's.

References

See a tutorial/vignette for alm at http://ropensci.org/tutorials/alm_tutorial.html

See Also

alm_plot

Examples

```
## Not run:
# The default call with either doi, pmid, pmcid, wos, scp, or url without specifying
# an argument for info
alm_ids(doi="10.1371/journal.pone.0029797")

# Details for a single DOI
out <- alm_ids(doi='10.1371/journal.pone.0029797', info='detail')
out
## totals
out$data$info
## history
out$data$sum_metrics

# A single PubMed ID (pmid)
alm_ids(pmid=22590526)

# A single PubMed Central ID (pmcid)
alm_ids(pmcid=212692, info='summary')

# A single Web of Science ID (wos)
```

```
alm_ids(wos="000268452400005")

# A single Scopus ID (scp)
alm_ids(scp="68049122102")

# A single Canonical URL (url)
alm_ids(url="http://www.plosmedicine.org/article/info:doi/10.1371/journal.pmed.1000097")

# Provide more than one DOI
dois <- c('10.1371/journal.pone.0001543','10.1371/journal.pone.0040117',
'10.1371/journal.pone.0029797','10.1371/journal.pone.0039395')
out <- alm_ids(doi=dois)
out$data[[1]] # get data for the first DOI

# Search for DOI's, then feed into alm
library('rplos')
dois <- searchplos(q='evolution', fl='id',
   fq=list('-article_type:correction','doc_type:full'), limit = 250)
out <- alm_ids(doi=dois$data$id)
lapply(out, head)

alm_ids(dois$data$id[1:5], source_id = "facebook")

sources <- c("facebook","twitter","mendeley","reddit","scopus","wikipedia")
lapply(sources, function(x) alm_ids(dois$data$id[1:5], source_id = x))

# Provide more than one pmid
pmids <- c(19300479, 19390606, 19343216)
out <- alm_ids(pmid=pmids)
out$data[[3]] # get data for the third pmid

# Getting data for a specific source_id
alm_ids(doi='10.1371/journal.pone.0035869', source_id='mendeley')
alm_ids(doi='10.1371/journal.pone.0035869', source_id='twitter')
alm_ids(doi='10.1371/journal.pone.0035869', source_id='counter', info='detail')
## fails if more than one source_id given
# alm_ids(doi='10.1371/journal.pone.0035869', source_id=c('twitter','facebook'))

# Get detailed totals output
alm_ids(doi='10.1371/journal.pone.0035869', total_details=TRUE)

# Get summary metrics by day
alm_ids(doi='10.1371/journal.pone.0036240', sum_metrics='day')

# Get summary metrics by month
alm_ids(doi='10.1371/journal.pone.0036240', sum_metrics='month')

# Get summary metrics by year
alm_ids(doi='10.1371/journal.pone.0036240', sum_metrics='year')

# Get data by source_id
alm_ids(source_id='crossref')
alm_ids(source_id='twitter')
```

```
# Curl debugging
library('httr')
alm_ids(doi="10.1371/journal.pone.0029797", config=verbose())
dois <- c('10.1371/journal.pone.0001543','10.1371/journal.pone.0040117',
  '10.1371/journal.pone.0029797','10.1371/journal.pone.0039395')
alm_ids(doi=dois, config=progress())

# Data from other sources
## Crossref article data
### Pass in a different URL - no key needed
api_url <- "http://det.labs.crossref.org/api/v5/articles"
alm_ids(doi='10.1371/journal.pone.0086859', api_url = api_url)
alm_ids(doi='10.11646/zootaxa.3618.1.1', api_url = api_url)
alm_ids(doi='10.1016/j.jep.2013.06.007', api_url = api_url)
alm_ids(doi='10.1111/j.1756-1051.2012.00099.x', api_url = api_url)

## Public Knowledge Project article data
### pass in a different url - an API key needed
api_url <- 'http://pkp-alm.lib.sfu.ca/api/v5/articles'
alm_ids(doi='10.3402/gha.v7.23554', api_url = api_url, key = getOption("pkpalmkey"))

## eLife
### pass in a different url - no key needed
api_url <- 'http://lagotto.svr.elifesciences.org/api/v5/articles'
alm_ids(doi='10.7554/eLife.00471', api_url = api_url)

## End(Not run)
```

`alm_plot` *Plot results of a call to the alm function.*

Description

Plot results of a call to the alm function.

Usage

```
alm_plot(dat)
```

Arguments

`dat`	Output from `alm_ids` (character)

Value

A ggplot2 bar plot for ‘totalmetrics‘ or line plot for ‘history‘.

References

See a tutorial/vignette for alm at http://ropensci.org/tutorials/alm_tutorial.html

See Also

alm_ids which is required to use this function.

Examples

```
## Not run:
out <- alm_ids(doi='10.1371/journal.pone.0001543', info='detail')
alm_plot(out)
# works from info=totals too
out <- alm_ids(doi='10.1371/journal.pone.0001543', info='totals')
alm_plot(out)

## End(Not run)
```

---

alm_requests *Retrieve API requests data*

Description

Retrieve API requests data

Usage

```
alm_requests(key, url = "http://alm.plos.org/api/v5/api_requests", ...)
```

Arguments

key	API key. Required.
url	URL for the api requests endpoint. Required.
...	Curl args to GET

Details

This function requires authentication via an API key.

Examples

```
## Not run:
out <- alm_requests()
out$meta
dat <- out$data
head( dat )
dat[ dat$db_duration > 100, ]

## End(Not run)
```

`alm_signposts` *Retrieve PLOS article-level metrics signposts.*

Description

This includes:

- viewed: counter, and pmc (PLOS only)
- saved: mendeley, and citeulike
- discussed: facebook, and twitter at PLOS
- cited: crossref, and scopus at PLOS

Usage

```
alm_signposts(doi = NULL, pmid = NULL, pmcid = NULL, wos = NULL,
  scp = NULL, url = NULL, source_id = NULL, key = NULL,
  api_url = "http://alm.plos.org/api/v5/articles", ...)
```

Arguments

`doi`	Digital object identifier for an article in PLoS Journals (character)
`pmid`	PubMed object identifier (numeric)
`pmcid`	PubMed Central object identifier (numeric)
`wos`	Web of Science identifier (character)
`scp`	Scopus identifier (character)
`url`	Canonical URL (character)
`source_id`	(character) Name of source to get ALM information for. One source only. You can get multiple sources via a for loop or lapply-type call.
`key`	your PLoS API key, either enter, or loads from .Rprofile (character)
`api_url`	API endpoint, defaults to http://alm.plos.org/api/v3/articles (character)
`...`	optional additional curl options (debugging tools mostly)

Details

This is just a wrapper around the function `alm_ids`, forcing info="summary", then coercing signposts data to a data.frame.

Value

A data.frame of the signpost numbers for the searched object, and DOIs.

References

See a tutorial/vignette for alm at `http://ropensci.org/tutorials/alm_tutorial.html`

See Also

alm_ids, plot_signposts

Examples

```
## Not run:
# The default call with either doi, pmid, pmcid, wos, scp, or url without specifying
# an argument for info
alm_signposts(doi="10.1371/journal.pone.0029797")

# Many DOIs
dois <- c('10.1371/journal.pone.0001543','10.1371/journal.pone.0040117',
'10.1371/journal.pone.0029797','10.1371/journal.pone.0039395')
alm_signposts(doi=dois)

# A single PubMed ID (pmid)
alm_signposts(pmid=22590526)

# A single PubMed Central ID (pmcid)
alm_signposts(pmcid=212692)

# A single PubMed Central ID (pmcid)
alm_signposts(source_id = "crossref")

# Curl debugging
library('httr')
alm_signposts(pmid=22590526, config=verbose())

## End(Not run)
```

---

alm_sources *Retrieve PLoS article-level metrics (ALM) by source.*

Description

See details for more information.

Usage

```
alm_sources(source_id = "crossref", info = "totals", key = NULL,
  total_details = FALSE, sum_metrics = NULL, limit = 50, page = 1,
  url = "http://alm.plos.org/api/v5/articles", ...)
```

Arguments

source_id	(character) Name of source to get ALM information for. One source only. You can get multiple sources via a for loop or lapply-type call.

info	One of totals, summary, or detail (default totals + sum_metrics data in a list). Not specifying anything (the default) returns data.frame of totals across data providers. (character)
key	(character) Your API key, either enter, or loads from .Rprofile. Only required for PKP source, not the others.
total_details	If FALSE (the default) the standard totals data.frame is returned; if TRUE, the totals data is in a wide format with more details about the paper, including publication date, title, etc. If you set this to TRUE, the output should no longer with with alm_plot.
sum_metrics	Just like the output you get from setting info='totals', you can get summary metrics by day (sum_metrics='day'), month (sum_metrics='month'), or year (sum_metrics='year').
limit	(integer) Number from 1 to infinity. This doubles as the rows parameter, which is what's called internally to the API service. The max results per page is 50, so if you use a value > 50, then we essentially loop through to get all the results you want.
page	(integer) Number from 1 to infinity.
url	API endpoint, defaults to http://alm.plos.org/api/v3/articles (character)
...	optional additional curl options (debugging tools mostly)

References

See a tutorial/vignette for alm at http://ropensci.org/tutorials/alm_tutorial.html

Examples

```
## Not run:
alm_sources()
alm_sources(source_id='mendeley')
alm_sources(source_id='scopus', info='summary')
lapply(c('mendeley','twitter'), alm_sources, limit = 2)
alm_sources(source_id='mendeley', limit=2)
alm_sources(source_id='mendeley', limit=2, page=2)
alm_sources(source_id='mendeley', limit=200)

alm_sources(source_id='mendeley', info='summary')

## End(Not run)
```

---

alm_status *Check status of an ALM service.*

Description

Check status of an ALM service.

Usage

```
alm_status(key = NULL, url = "http://alm.plos.org/api/v5/status", ...)
```

Arguments

key	(character) Your API key, either enter, or loads from .Rprofile. Only required for PKP source, not the others.
url	API endpoint, defaults to http://alm.plos.org/api/v5/status (character)
...	optional additional curl options (debugging tools mostly)

Examples

```
## Not run:
alm_status()

## End(Not run)
```

---

alm_title *Get title of article by inputting the doi for the article.*

Description

Get title of article by inputting the doi for the article.

Usage

```
alm_title(doi = NULL, pmid = NULL, pmcid = NULL, wos = NULL,
  scp = NULL, url = NULL, key = NULL,
  api_url = "http://alm.plos.org/api/v5/articles", ...)
```

Arguments

doi	Digital object identifier for an article in PLoS Journals (character)
pmid	PubMed object identifier (numeric)
pmcid	PubMed Central object identifier (numeric)
wos	Web of Science identifier (character)
scp	Scopus identifier (character)
url	Canonical URL (character)
key	your PLoS API key, either enter, or loads from .Rprofile (character)
api_url	API endpoint, defaults to http://alm.plos.org/api/v3/articles (character)
...	optional additional curl options (debugging tools mostly)

Value

Title of article, in xml format.

References

See a tutorial/vignette for alm at http://ropensci.org/tutorials/alm_tutorial.html

Examples

```
## Not run:
alm_title(doi='10.1371/journal.pbio.0000012')
dois <- c('10.1371/journal.pone.0026871','10.1371/journal.pone.0048868',
   '10.1371/journal.pone.0048705','10.1371/journal.pone.0048731')
alm_title(doi=dois)

## End(Not run)
```

`plot_density` *Density and histogram plots from PLOS Article Level Metrics data*

Description

Density and histogram plots from PLOS Article Level Metrics data

Usage

```
plot_density(input, source = "scopus_total", color = "#1447f2",
  title = "", description = "", plot_type = "density")
```

Arguments

`input`	A data.frame, usuaally from a call to `link{alm}`.
`source`	Data source (column) to plot. Can be a single element, or a character vector.
`color`	Color of the density plot and the title. Can be a hex color or rgb, etc.
`title`	Title for the plot, in top matching the color of the density plot.
`description`	Optional description, subtending the title.
`plot_type`	Type of plot, one of density (default) or histogram.

Author(s)

Martin Fenner, mfenner@plos.org, modified by Scott Chamberlain

References

See a tutorial/vignette for alm at http://ropensci.org/tutorials/alm_tutorial.html

Examples

```
## Not run:
library('rplos'); library('plyr')
dois <- searchplos(q='*:*', fl="id",
   fq=list(
     'cross_published_journal_key:PLoSONE',
     'doc_type:full',
     'publication_date:[2010-01-01T00:00:00Z TO 2010-12-31T23:59:59Z]',
     '-article_type:correction'),
   limit=50)
dois <- dois$data$id[!grepl("annotation", dois$data$id)]
alm <- alm_ids(doi=dois, total_details=TRUE)
alm <- ldply(alm$data, data.frame)
plot_density(alm)
plot_density(alm, color="#DCA121")
plot_density(alm, title="Scopus citations from 2010")
plot_density(alm, title="Scopus citations from 2010", description="Probablity of
   X number of citations for a paper")
plot_density(alm, description="Probablity of X number of citations for a paper")
plot_density(alm, source="crossref_total")
plot_density(alm, source="twitter_total")
plot_density(alm, source="counter_total")
plot_density(alm, source=c("counter_total","facebook_likes"))
plot_density(alm, source=c("counter_total","facebook_likes"))
plot_density(alm, source=c("counter_total","facebook_likes", "twitter_total"))
plot_density(alm, source=c("counter_total","crossref_total","twitter_total"),
   color=c("#DBAC6A", "#E09B33", "#A06D34"))
plot_density(alm, source=c("counter_total","crossref_total",
   "twitter_total"))
plot_density(alm, source=c("counter_total","crossref_total"),
   title="Counter and Crossref")
plot_density(alm, source=c("counter_total","crossref_total",
   "twitter_total"), color=c("#83DFB4","#EFA5A5","#CFD470"))

## End(Not run)
```

`plot_signposts` *Plot PLOS article-level metrics data.*

Description

This can be used in conjuction with the function `alm_signposts`.

Usage

```
plot_signposts(input)
```

Arguments

`input`	A data.frame from a search from the `alm_signposts` function

Details

Note that DOIs are the unit of replication of each study. When plotting, if the prefix is common among all DOIs, then just the end of the DOI, the numeric part is printed to make plots less ugly.

References

See a tutorial/vignette for alm at http://ropensci.org/tutorials/alm_tutorial.html

See Also

alm_signposts

Examples

```
## Not run:
# Plot data from a single identifier gives a bar chart
dat <- alm_signposts(doi="10.1371/journal.pone.0029797")
plot_signposts(dat)

# Plot data from many identifiers gives a line chart
dois <- c('10.1371/journal.pone.0001543','10.1371/journal.pone.0040117',
   '10.1371/journal.pone.0029797','10.1371/journal.pone.0039395')
dat <- alm_signposts(doi=dois)
plot_signposts(dat)

# software lagotto instance
urls <- c("https://github.com/najoshi/sickle","https://github.com/lh3/wgsim",
   "https://github.com/jstjohn/SeqPrep")
dat <- alm_signposts(url = urls, api_url = "http://software.lagotto.io/api/v5/articles")
plot_signposts(dat)

# scopus ids
ids <- c(68049122102, 14044251458, 48349097292, 28444460441)
dat <- alm_signposts(scp = ids)
plot_signposts(dat)

## End(Not run)
```

Index