



CARD.com

Sponsored · 🌐

👍 Like Page

Free webinar on how to create, manage and optimize Facebook ads from R.



FREE WEBINAR

Manage Facebook ads from R

March 9th, 2016, 11-11:30am PST

Webinar on FB Ads API & R

WWW.DOMINODATALAB.COM

Sign Up

Domino Data Lab webinar

Managing and Optimizing Facebook Ad Campaigns with R



CARD.COM

Gergely Daroczi

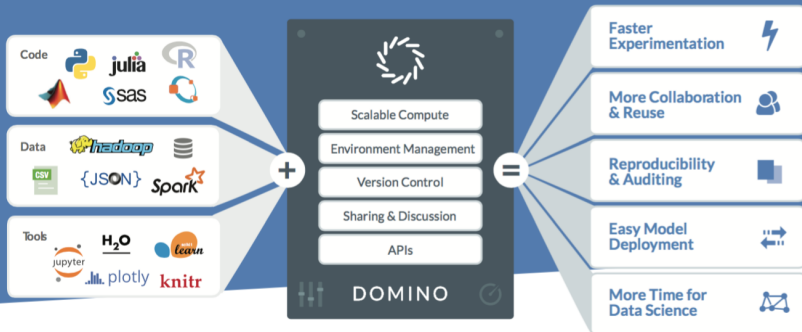
@daroczig

March 9 2016



DOMINO
DATA LAB

Domino Enterprise Data Science Platform



**PRODUCTIVITY
FOR INDIVIDUALS**

**COLLABORATION
FOR TEAMS**

**CENTRALIZATION
OF KNOWLEDGE**

www.dominodatalab.com



5



Thanks for you kind words and great question. There's a not yet well documented feature in `pander`, but you can also pass an R function as the [default table alignment](#). Quick demo:

```
> panderOptions('table.alignment.default',
+   function(df) ifelse(sapply(df, is.numeric), 'right', 'left'))
> pander(data.frame(
+   name      = letters[1:3],
+   size      = 1:3,
+   we.have.dates = Sys.Date() - 1:3
+ ))
```

```
-----
name      size we.have.dates
-----
a          1 2014-11-18
b          2 2014-11-17
c          3 2014-11-16
-----
```

So the trick here is to define a function which takes only one argument to be analysed, and it returns the vector of column alignment parameters.

share edit delete flag

answered Nov 19 '14 at 10:28



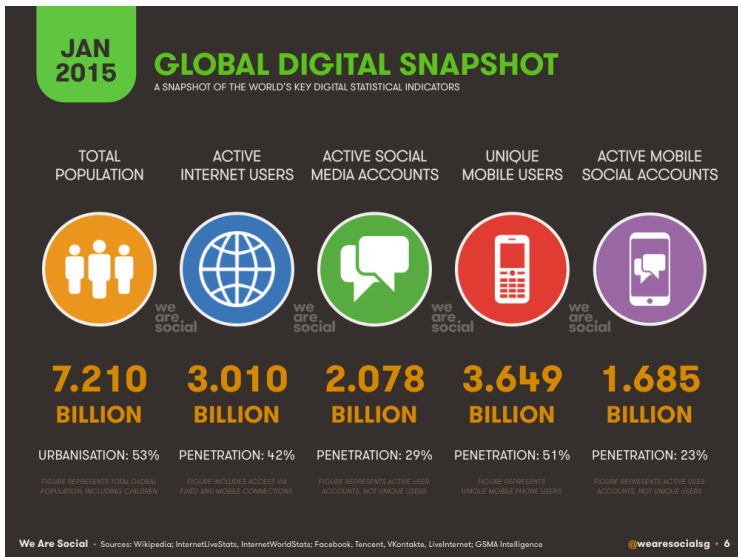
[daroczig](#)

14.4k ● 5 ● 38 ● 73

Why isn't this default? Was pander created by "marketing people"? – [Waldir Leoncio](#) May 29 at 20:00

When to Advertise on Facebook







Gergely Daróczy @daroczig · Apr 11

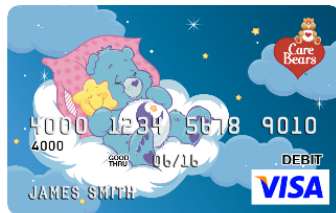
Just received my "I ♥ R" prepaid debit card from @CARD. Will be fun to use this #rstats designed card at #user2015 :)

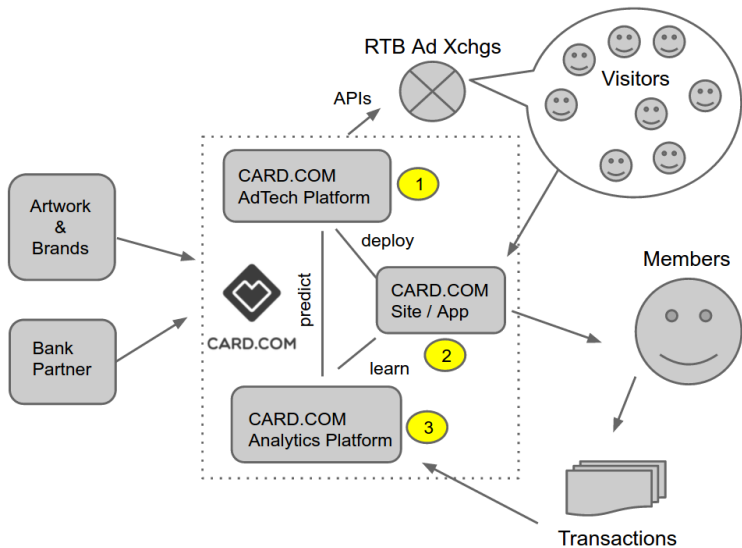


RETWEETS
10

FAVORITES
16







- Google knows what you are searching for
- Amazon knows what you are in the market for
- Facebook knows what you like

- Google knows what you are searching for

```
devtools::install.github('jburkhardt/RAdwords')
```

- Amazon knows what you are in the market for

```
NULL
```

- Facebook knows what you like

```
devtools::install.github('cardcorp/fbRads')
```

- Google knows what you are searching for

```
devtools::install.github('jburkhardt/RAdwords')
```

- Amazon knows what you are in the market for

```
NULL
```

- Facebook knows what you like

```
devtools::install.github('cardcorp/fbRads')
```

- This info can be is used to advertise to you ...

- Google knows what you are searching for

```
devtools::install.github('jburkhardt/RAdwords')
```

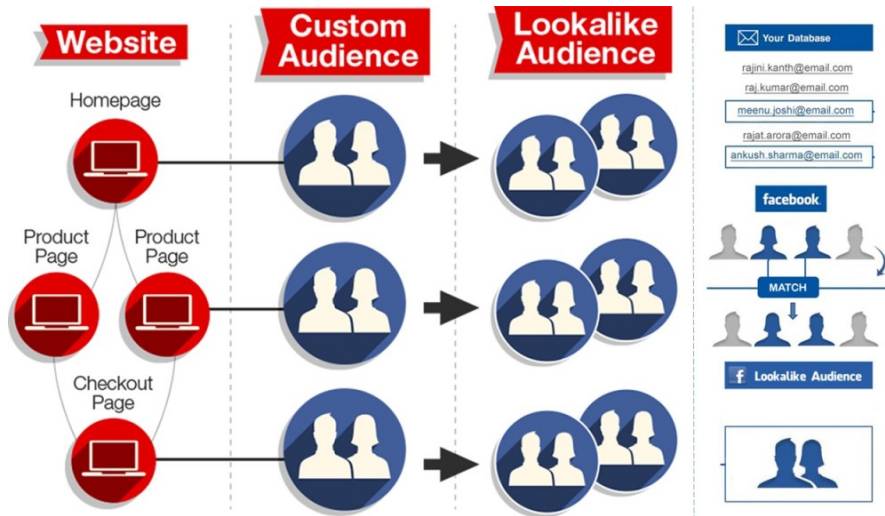
- Amazon knows what you are in the market for

NULL

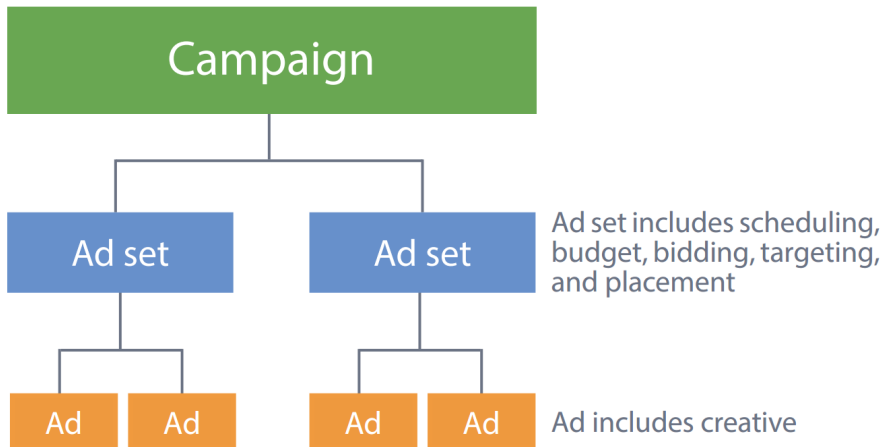
- Facebook knows what you like

```
devtools::install.github('cardcorp/fbRads')
```

- This info can be is used to advertise to you ...
- ... without much privacy concerns. How?



Source: adparlor.com



Source: Facebook Marketing API docs

<https://developers.facebook.com/docs/marketing-api/changelog>

Version	Path	Date Introduced	Available Until
v2.5	/v2.5/{object}	Oct 7th, 2015	
v2.4	/v2.4/{object}	July 8th, 2015	April, 2016
v2.3	/v2.3/{object}	March 25th, 2015	October 7th, 2015
v2.2	/v2.2/{object}	October 30th, 2014	July 8th, 2015
v2.1	/v2.1/{object}	October 1st, 2014	March 11th, 2015
v2.0	/v2.0/{object}	October 1st, 2014	March 11th, 2015
v1.0	/v1.0/{object}	October 1st, 2014	March 11th, 2015
non-versioned path	/ {object}		March 11th, 2015

- new API version every 6 months

2014:

- campaigns: start/end date, budget
- ads: targeting, placement, creative
- Ad Report Stats

2015:

- campaigns [`campaign_group`]: start/end date
- adset [`campaign`]: budget, targeting, placement
- ads [`ad_group`]: creative
- Insights API

2016:

- Consistent naming between API and UI



CARD.com

Sponsored · 

 Like Page

Free webinar on how to create, manage and optimize Facebook ads from R.



FREE WEBINAR

Manage Facebook ads from R

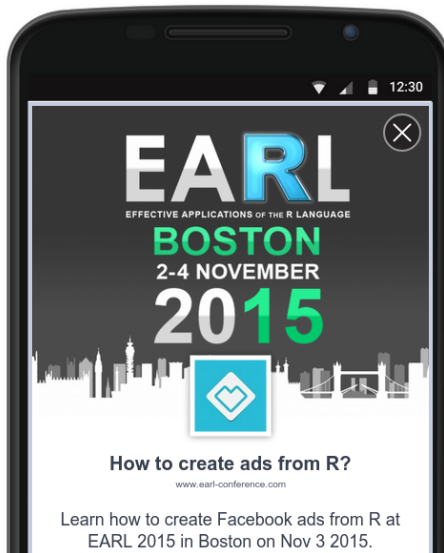
March 9th, 2016, 11-11:30am PST

Webinar on FB Ads API & R

WWW.DOMINODATALAB.COM

Sign Up

Has anyone seen this?





How to create ads from R?

`user2015.math.aau.dk`

Learn how to create Facebook ads from R
at a contributed talk at the useR! 2015
conference

Get R package developer e-mails

Maintainer	Package	Version	r-devel Linux x86_64 (Debian GCC)	r-devel Linux x86_64 (Fedora Clang)	r-devel Linux x86_64 (Fedora GCC)	r-devel OS X x86_64 (Clang)	r-devel Windows x86+x86_64	r-patched Linux x86_64	r-patched Solaris sparc	r-patched Solaris x86	r-release Linux x86_64	r-release OS X x86_64 (Mavericks)	r-release Windows x86+x86_64	r-older Windows x86+x86_64	Priority
Daniel Gerhard <dgerhard at gmail.com>	gorie	0.0-8	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	OK	
	mpcprofile	0.2-1	NOTE	NOTE	NOTE	NOTE	NOTE	OK	OK	OK	OK	NOTE	OK	OK	
Zheng-Sengui <1225620446 at qq.com>	NLRoot	1.0	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	
Xiao-Ping You <13268259225 at 163.com>	XHWE	1.0	NOTE	NOTE	NOTE	NOTE	NOTE	OK	OK	OK	OK	OK	OK	OK	
huma Ihnatoya <184415 at mail.muni.cz>	MAMA	2.2.1	NOTE	NOTE	NOTE	NOTE	NOTE*	NOTE	NOTE	NOTE	NOTE	WARN	NOTE*	NOTE*	
Mikiya TANIZAWA <1970mix at gmail.com>	mspackR	1.1	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	
Chengchun Shi <3100102177 at zju.edu.cn>	BayesSAE	1.0-1	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	
Chengfeng Liu <478996060 at qq.com>	hypothesetest	1.0	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	OK	OK	NOTE	OK	OK	OK	
H.R.Law <4islands at gmail.com>	EW	1.1	NOTE	NOTE	NOTE	NOTE	NOTE	OK	OK	OK	OK	OK	OK	ERROR	
Greg Snow <538280 at gmail.com>	TeachingDemos	2.10	NOTE	NOTE	NOTE	NOTE	OK	NOTE	NOTE	NOTE	NOTE	NOTE	OK	OK	
Zeng RuTong <670273197 at qq.com>	clusterfend	1.4	NOTE	NOTE	NOTE	NOTE	NOTE	OK	OK	OK	OK	OK	OK	OK	
Ting Yang <707237077 at qq.com>	signmedian.test	1.5.1	NOTE	NOTE	NOTE	NOTE	NOTE	OK	OK	OK	OK	OK	OK	OK	
Chih-Yi Chien <92354503 at nctu.edu.tw>	MAYTgsa	1.3	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	OK	
Artem Klevtsov <a.a.klevtsov at gmail.com>	RG4	0.4.0												WARN	
	RG4	0.4.1	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK		
Anup Amatya <aamatya at nmsu.edu>	BinNor	2.0	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	OK	
	MultiOrd	2.1	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	OK	
	OrdNor	2.0	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	
	PoisNor	1.0	NOTE	NOTE	NOTE	NOTE	NOTE	OK	OK	OK	OK	OK	OK	OK	
Amelie Anota <aanota at chu-besancon.fr>	QoLR	1.0.2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	
Alberto Arcagni <a.arcagni at campus.unimib.it>	sIRF	1.1.1	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	OK	
Aaron Robotham <aaron.robtham at syva.edu.au>	celestial	1.3	NOTE	NOTE	NOTE	NOTE	NOTE	OK	OK	OK	OK	OK	OK	OK	
	magixaxis	1.9.4	NOTE	NOTE	NOTE	NOTE	NOTE	OK	OK	OK	OK	OK	OK	OK	
	snapshot	0.1.2	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	OK	
	apbsplot	1.5	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	
	lged	0.1.1	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	
Claus Dethlefsen <aas.claus.dethlefsen at gmail.com>	giRaph	0.1.2	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	
	mathgraph	0.9-11	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	OK	
Adrian Barnett <a.barnett at qut.edu.au>	season	0.3-5	NOTE	NOTE	NOTE	NOTE	NOTE	OK	OK	OK	OK	OK	OK	OK	
Abdelmonem Amer Desouki <abdelmonem.amer at uni-duesseldorf.de>	xybliceBA	2.0.0	NOTE	NOTE	NOTE	NOTE	NOTE	OK	NOTE	NOTE	OK	NOTE	NOTE	NOTE	
	xybicycleFreeFlux	1.0.1	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	
	xybliceFBA	1.0.2	NOTE	NOTE	NOTE	NOTE	NOTE	OK	NOTE	NOTE	OK	NOTE	NOTE	NOTE	
	xybidiDynFBA	0.0.2	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	NOTE	
Alfonso Benitez-Paez <abenitez at iata.csic.es>	permubiome	1.0	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	WARN	
Art B. Friedman <abfriedman at gmail.com>	bootLR	1.0	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	

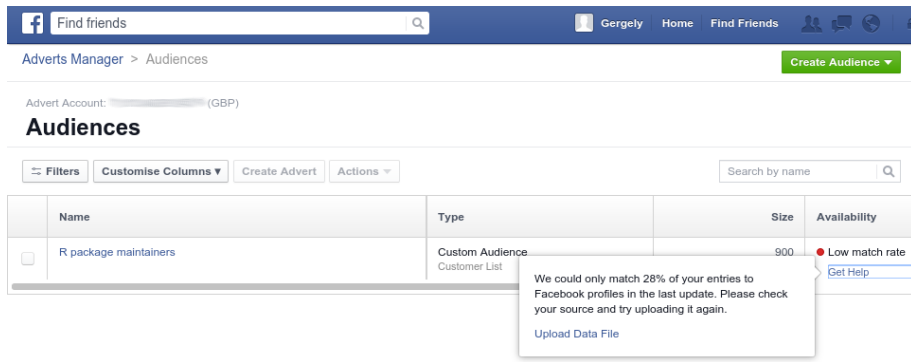
```
> url <- 'http://cran.r-project.org/web/checks/check_summary.html'
> packages <- XML::readHTMLTable(url, which = 2)
> mails <- sub('.*<(.*)>', '\\1', packages$' Maintainer')
> mails <- sub(' at ', '@', mails)
```

```
> url <- 'http://cran.r-project.org/web/checks/check_summary.html'
> packages <- XML::readHTMLTable(url, which = 2)
> mails <- sub('.*<.*>', '\\1', packages$'Maintainer')
> mails <- sub(' at ', '@', mails)
```

```
> tail(sort(table(mails)))
## edd@debian.org (35)           # Dirk Eddelbuettel
## Kurt.Hornik@R-project.org (29) # Kurt Hornik
## myrmecocystus@gmail.com (24)  # Scott Chamberlain
## maechler@stat.math.ethz.ch (24) # Martin Maechler
## pgilbert.ttv9z@ncf.ca (22)    # Paul Gilbert
```

```
> length(unique(mails))
## 4023
```

```
> tail(sort(table(sub('.*@', '', mails))))
## gmail.com (1778)
## R-project.org (84)
## edu
```



The screenshot shows the Facebook Adverts Manager interface. At the top is a navigation bar with a search bar, user profile, and links to Home and Find Friends. Below this is a breadcrumb trail: Adverts Manager > Audiences. A green button labeled 'Create Audience' is in the top right. The main section is titled 'Audiences' and shows the Advertiser Account as 'Gergely Daroczi' (GBP). Below the title are buttons for 'Filters', 'Customise Columns', 'Create Advert', and 'Actions'. A search bar labeled 'Search by name' is on the right. A table lists the audience 'R package maintainers' with a size of 900 and a 'Low match rate' status. A tooltip message states: 'We could only match 28% of your entries to Facebook profiles in the last update. Please check your source and try uploading it again.' with a link to 'Upload Data File'.

	Name	Type	Size	Availability
<input type="checkbox"/>	R package maintainers	Custom Audience Customer List	900	Low match rate Get Help

28 % match: only 900 accounts for 6,000+ R packages

Get the location of the archives:

```
> url <- 'https://stat.ethz.ch/pipermail/r-help/'
```

We need Rcurl for HTTPS:

```
> library(Rcurl)
```

Get URL of all archive files:

```
> R.help.toc <- htmlParse(getURL(url))
> R.help.archives <- unlist(
+   xpathApply(R.help.toc, "///table//td[3]/a", xmlAttrs),
+   use.names = FALSE)
```

Download archive files:

```
> dir.create('r-help')
> for (f in R.help.archives)
+   download.file(url = paste0(url, f),
+                 file.path('help-r', f), method = 'curl')
```

Regular expression matching date format in “From” lines:

```
> dateregex <- paste('[A-Za-z]{3} [A-Za-z]{3} [0-9]{1,2}',  
+                    '[0-9]{2}:[0-9]{2}:[0-9]{2} [0-9]{4}')
```

grep for lines matching the From field:

```
> mails <- system(paste0(  
+   "zgrep -E '^From .* at .* ",  
+   dateregex,  
+   "' ./help-r/*.txt.gz"),  
+                 intern = TRUE)
```

Extract e-mail addresses from these lines:

```
> mails <- sub('.*From ', '', mails)  
> mails <- sub(paste0('[ ]*', dateregex, '$'), '', mails)  
> mails <- sub(' at ', '@', mails)
```

```
> length(mails)
266449

> head(sort(table(mails), decreasing = TRUE))
      ripley@stats.ox.ac.uk      dwinsemius@comcast.net
                        8611                        7064
ggrothendieck@gmail.com p.dalgaard@biostat.ku.dk
                        5386                        3243
      jholtman@gmail.com      smartpink111@yahoo.com
                        3193                        2999
```

```
> grep('Brian( D)? Ripley', names(table(mails)), value = TRUE)
[1] "Brian D Ripley"
[2] "Brian D Ripley [mailto:ripley at stats.ox.ac.uk]"
[3] "Brian Ripley"
[4] "Brian Ripley <ripley at stats.ox.ac.uk>"
[5] "Prof Brian D Ripley"
[6] "Prof Brian D Ripley [mailto:ripley at stats.ox.ac.uk]"
[7] "          Prof Brian D Ripley <ripley at stats.ox.ac.uk>"
[8] "\"Prof Brian D Ripley\" <ripley at stats.ox.ac.uk>"
[9] "Prof Brian D Ripley <ripley at stats.ox.ac.uk>"
[10] "Prof Brian Ripley"
[11] "Prof. Brian Ripley"
[12] "Prof Brian Ripley [mailto:ripley at stats.ox.ac.uk]"
[13] "Prof Brian Ripley [mailto:ripley at stats.ox.ac.uk] "
[14] "          \tProf Brian Ripley <ripley at stats.ox.ac.uk>"
[15] "  Prof Brian Ripley <ripley at stats.ox.ac.uk>"
[16] "\"Prof Brian Ripley\" <ripley at stats.ox.ac.uk>"
[17] "Prof Brian Ripley<ripley at stats.ox.ac.uk>"
[18] "Prof Brian Ripley <ripley at stats.ox.ac.uk>"
[19] "Prof Brian Ripley [ripley at stats.ox.ac.uk]"
[20] "Prof Brian Ripley <ripley at toucan.stats>"
[21] "Professor Brian Ripley"
[22] "r-help-bounces at r-project.org [mailto:r-help-bounces at r-project.org] On B
[23] "r-help-bounces at stat.math.ethz.ch [mailto:r-help-bounces at stat.math.ethz.

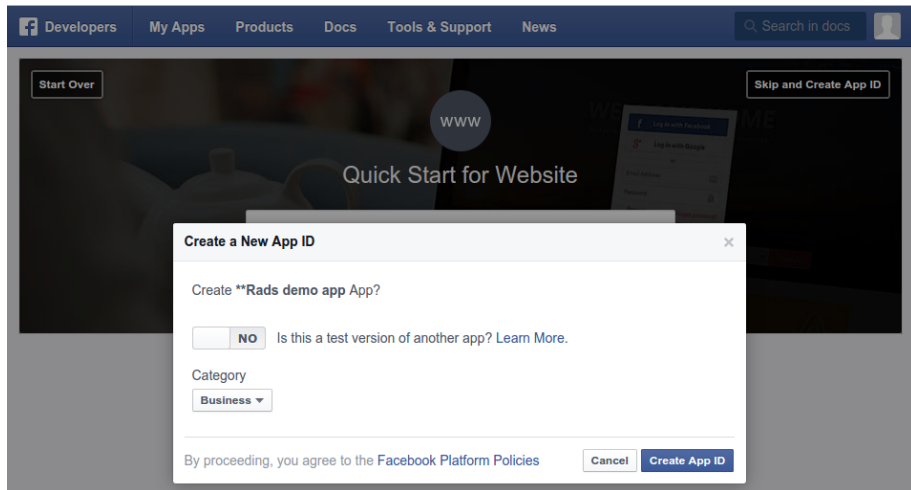
```

```
> length(mails)
266449

> head(sort(table(mails), decreasing = TRUE))
      ripley@stats.ox.ac.uk      dwinsemius@comcast.net
                        8611                        7064
ggrothendieck@gmail.com p.dalgaard@biostat.ku.dk
                        5386                        3243
      jholtman@gmail.com      smartpink111@yahoo.com
                        3193                        2999

> length(unique(mails))
29266

> 29266 > 4023
TRUE \o/
```



<https://developers.facebook.com/apps/>

Create a token:

```
> library(httr)
> app <- oauth_app('facebook', 'your_app_id', 'your_app_secret')
> Sys.setenv('HTTR_SERVER_PORT' = '1410/')
> tkn <- oauth2.0_token(
+   oauth_endpoints('facebook'), app, scope = 'ads_management',
+   type = 'application/x-www-form-urlencoded')
> tkn <- tkn$credentials$access_token
```

Save this secret token (never commit to git repository) and load it in any later session:

```
> saveRDS(tkn, 'token.rds')
> tkn <- readRDS('token.rds')
```

Initialize connection to Facebook Marketing API:

```
> devtools::install_packages('cardcorp/fbRads')  
> library(fbRads)  
> fbad_init(fid, tkn)
```



```
> aud_id <- fbad_create_audience(name = 'R-help posters',  
+   title = 'Unique e-mail addresses in R-help 1997-2015')
```

Reading audience info:

```
> fbad_read_audience(audience_id = aud_id,  
+   fields = 'approximate_count')  
20
```

Adding e-mails to audience (be patient):

```
> fbad_add_audience(audience_id = aud_id,  
+   schema = 'EMAIL', hashes = mails)  
> fbad_read_audience(audience_id = aud_id,  
+   fields = 'approximate_count')  
8700
```

Load the number of attendees per country:

```
> url <- 'http://rapporter.net/custom/R-activity/data/Rstats_2015.csv'
> library(data.table)
> RpC <- fread(url)
> conference_countries <- RpC[user_all > 0, ]
```

Create a lookalike audience for each country with at least one useR! conference attendee:

```
> aud_ids <- sapply(1:nrow(conference_countries), function(i) {
+
+   try(fbad_create_lookalike_audience(
+     name           = paste('R-help posters in',
+                           conference_countries[i, NAME]),
+     origin_audience_id = aud_id,
+     ratio            = 0.01,
+     country          = toupper(conference_countries[i, ISO2C]))
+
+   Sys.sleep(20)
+ })
```

Get the approximate count of each lookalike audience:

```
> lookalikes[!is.na(audience),  
+   size := fbad_read_audience(audience, 'approximate_count')[[1]],  
+   by = country]
```

```
> lookalikes[!is.na(audience), c('country', 'size'), with = FALSE]
```

	country	size		country	size
1:	Australia	173000	13:	Ireland	32800
2:	Austria	41500	14:	Italy	336200
3:	Belgium	72400	15:	Latvia	7800
4:	Brazil	1280400	16:	Mexico	758100
5:	Canada	253100	17:	Netherlands	110900
6:	Colombia	308200	18:	New Zealand	34500
7:	Faroe Islands	400	19:	Norway	36500
8:	France	392900	20:	Singapore	257000
9:	Germany	347700	21:	Slovenia	11200
10:	Greece	59900	22:	Spain	284200
11:	Hungary	61500	23:	Switzerland	43100
12:	India	2042000	24:	United Kingdom	478700
			25:	United States	2483200

```
> campaign <- fbad_create_campaign(  
+   name      = 'Promote the Domino Data Lab webinar',  
+   objective = 'LINK_CLICKS')  
> fbad_read_campaign(id = campaign,  
+   fields = c('id', 'buying_type', 'name', 'objective'))  
$id  
[1] "*****"  
  
$buying_type  
[1] "AUCTION"  
  
$campaign_group_status  
[1] "ACTIVE"  
  
$objective  
[1] "NONE"  
  
$name  
[1] "Promote the Domino Data Lab webinar"
```

All valid lookalike audiences:

```
> target <- lookalikes[!is.na(audience)]  
> setnames(target, c('name', 'id'))
```

The original R-help posters list:

```
> target <- rbind(target, list('R-help poster list', id1))
```

The original R package developers list:

```
> target <- rbind(target, list('R pkg developers list', id2))
```

Prepare JSON list:

```
> target <- list(custom_audiences = target)
```

This is where we define the target and budget:

```
> adset <- fbad_create_adset(  
+   name = 'Promo budget for the Domino Data Lab webinar',  
+   campaign_id = campaign,  
+   billing_event = 'IMPRESSIONS',  
+   optimization_goal = 'REACH',  
+   bid_amount = 5,  
+   campaign_status = 'ACTIVE',  
+   lifetime_budget = 4200,  
+   end_time = as.numeric(as.POSIXct('2016-03-09')),  
+   targeting = target)
```

Get an image for the ad:

```
> img <- 'Domino-fbRads-webinar.png'  
> download.file('https://www.dominodatalab.com/resource/webinar/fbRads/Header.jpg',
```

Upload to Facebook:

```
> img <- fb_ad_create_image(img = img)
```

Take a note on the returned hash:

```
> str(img)  
List of 3  
 $ filename: chr "Domino-fbRads-webinar.png"  
 $ hash      : chr "0af72a03b5b91a7201718ec2f06513dd"  
 $ url       : chr "https://scontent.xx.fbcdn.net/hads-xfp1/t45.1600-4/124089"  
> img <- img$hash
```

```
> url <- file.path('https://www.dominodatalab.com/resource/web  
+                 'managing_optimizing_facebook_ad_campaigns_  
> (creative <- fbad_create_creative(  
+   name = 'How to create ads from R?',  
+   body = paste(  
+     'Free webinar on how to create, manage',  
+     'and optimize Facebook ads from R'),  
+   title      = 'How to create ads from R?',  
+   object_url = url,  
+   image_hash = img$hash))  
[1] 6036679347163
```

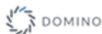



CARD.com

Sponsored · 

 Like Page

Free webinar on how to create, manage and optimize Facebook ads from R.



FREE WEBINAR

Manage Facebook ads from R

March 9th, 2016, 11-11:30am PST

Webinar on FB Ads API & R


WWW.DOMINODATALAB.COM

Sign Up




```
> ad <- fbad_create_ad(  
+   name      = 'An ad -- right from the R console',  
+   adset_id = adset,  
+   creative  = creative)
```



```
> taglines <- c('How to manage ads from R?',  
+               'How to optimize ads from R?')  
  
> for (tagline in taglines) {  
+  
+   ## create creative  
+   creative <- fb_ad_create_creative(  
+     name = tagline,  
+     body = paste(  
+       'Free webinar on how to create, manage',  
+       'and optimize Facebook ads from R'),  
+     title      = tagline,  
+     object_url = url,  
+     image_hash = img$hash)  
+  
+   ## create ad  
+   ad <- fb_ad_create_ad(  
+     name          = paste0(tagline),  
+     campaign_id = adset,  
+     creative      = creative)  
+ }
```

 Find friends

[Gergely](#)
[Home](#)
[Find Friends](#)

Adverts Manager

Account
Gergely Daróczi

Create an advert
Campaigns
 Pages
 Reports
 Audience Insights
 Settings
 Billing
 Conversion Tracking
 Power Editor
 Account History
 Audiences
 Help Centre
 Advertiser Support


Search your adverts

Home
 All Campaigns

Campaign
 Promote

Advert Set
 My budget for

How does this page work?
[Create Advert in Advert Set](#)

STATUS


DELIVERY
 Active

SPENT TODAY

LIFETIME SPENT

END DATE

RESULTS ?
 —

REACH ?

FREQUENCY ?




TOTAL SPENT ?

COST PER RESULT ?
 —

1 June 2015 - 30 June 2015

All Except Deleted
 Edit Adverts
 View Report
 View History

3 Results

Status ?	Advert ?	Delivery ?	Results ?	Cost ?	Reach ?	Frequency ?	Clicks ?	Click-Through Rate ?	Relevance Score ?	Spent Today	Total Spent ?	Max Bid ?	Avg. Price ?
		Active	Not Available	—					7/10			CPC	CPC
		Active	Not Available	—					7/10			CPC	CPC
		Active	Not Available	—					8/10			CPC	CPC

3 Results

[About](#)
[Create Advert](#)
[Create Page](#)
[Developers](#)
[Careers](#)
[Privacy](#)
[Cookies](#)
[AdChoices](#)
[Terms](#)
[Help](#)

Facebook © 2015
 English (UK)

```
> fb_insights(target = campaign, level = 'adgroup',  
+   fields = toJSON(c('reach', 'impressions', 'clicks')))
```

	reach	impressions	clicks	date_start	date_stop
1	16936	22369	119	2015-10-26	2015-11-02
2	7259	8318	29	2015-10-26	2015-11-02
3	19134	22539	63	2015-10-26	2015-11-02

```
> fb_insights(target = campaign, level = 'adgroup',  
+   fields = toJSON(c('reach', 'impressions', 'clicks'))))
```

	reach	impressions	clicks	date_start	date_stop
1	16936	22369	119	2015-10-26	2015-11-02
2	7259	8318	29	2015-10-26	2015-11-02
3	19134	22539	63	2015-10-26	2015-11-02

```
> fb_insights(target = campaign, level = 'adgroup',  
+   fields = toJSON(c('adgroup_name', 'cpc', 'cpp'))))
```

	adgroup_name	cpc	date_start	date_stop
1	Optimize ads	0.2344538	2015-10-26	2015-11-02
2	Manage ads	0.5031034	2015-10-26	2015-11-02
3	Create ads	0.4974603	2015-10-26	2015-11-02

```
> power.prop.test(p1 = 97 / 15682, p2 = 15 / 6672, power = 0.5, sig.level = 0.05)
```

Two-sample comparison of proportions power calculation

```
      n = 2081.102
     p1 = 0.006185436
     p2 = 0.002248201
sig.level = 0.05
  power = 0.5
alternative = two.sided
```

NOTE: n is number in *each* group

```
> fisher.test(data.frame(B = c(97, 15682), A = c(15, 6672)), conf.int = FALSE)
```

Fisher's Exact Test for Count Data

```
data: data.frame(B = c(97, 15682), A = c(15, 6672))
p-value = 6.811e-05
alternative hypothesis: true odds ratio is not equal to 1
sample estimates:
odds ratio
2.751109
```


Query sent to Facebook:

```
mystats <- fb_insights(date_preset = 'today', level = 'adgroup')
```

Query sent to Facebook:

```
mystats <- fb_insights(date_preset = 'today', level = 'adgroup')
```

Expected results:

```
$ adgroup_id      : chr  "... " "... " "... "  
$ campaign_id    : chr  "... " "... " "... "  
$ campaign_group_id : chr  "... " "... " "... "  
$ account_id     : chr  "... " "... " "... "  
$ frequency      : num  1.11 1.01 1.28  
$ impressions    : chr  "431" "280" "2735"  
$ reach          : int  390 277 2140  
$ cpc            : num  0.188 0.3 0.243  
$ cpm            : num  2.18 3.21 1.07  
$ cpp            : num  2.41 3.25 1.36  
$ ctr            : num  1.16 1.071 0.439
```

Query sent to Facebook:

```
mystats <- fb_insights(date_preset = 'today', level = 'adgroup')
```

Expected results:

```
$ adgroup_id      : chr  "... " "... " "... "  
$ campaign_id    : chr  "... " "... " "... "  
$ campaign_group_id : chr  "... " "... " "... "  
$ account_id     : chr  "... " "... " "... "  
$ frequency      : num  1.11 1.01 1.28  
$ impressions    : chr  "431" "280" "2735"  
$ reach         : int   390 277 2140  
$ cpc            : num   0.188 0.3 0.243  
$ cpm           : num   2.18 3.21 1.07  
$ cpp           : num   2.41 3.25 1.36  
$ ctr           : num   1.16 1.071 0.439
```

Response:

```
Error in fb_insights(date_preset = "today", level = "adgroup") (from fb_insights.R#9  
  (#100) Param level must be one of {ad, adset, campaign, account})
```

Query sent to Facebook:

```
mystats <- fb_insights(date_preset = 'today', level = 'ad')
```

Query sent to Facebook:

```
mystats <- fb_insights(date_preset = 'today', level = 'ad')
```

Expected results:

```
$ adgroup_id      : chr  "... " "... " "... "  
$ campaign_id     : chr  "... " "... " "... "  
$ campaign_group_id : chr  "... " "... " "... "  
$ account_id      : chr  "... " "... " "... "  
$ frequency       : num   1.11 1.01 1.28  
$ impressions      : chr  "431" "280" "2735"  
$ reach           : int   390 277 2140  
$ cpc              : num   0.188 0.3 0.243  
$ cpm              : num   2.18 3.21 1.07  
$ cpp              : num   2.41 3.25 1.36  
$ ctr              : num   1.16 1.071 0.439
```

Query sent to Facebook:

```
mystats <- fb_insights(date_preset = 'today', level = 'ad')
```

Expected results:

```
$ adgroup_id      : chr  "... " "... " "... "  
$ campaign_id    : chr  "... " "... " "... "  
$ campaign_group_id : chr  "... " "... " "... "  
$ account_id     : chr  "... " "... " "... "  
$ frequency      : num  1.11 1.01 1.28  
$ impressions     : chr  "431" "280" "2735"  
$ reach          : int  390 277 2140  
$ cpc            : num  0.188 0.3 0.243  
$ cpm            : num  2.18 3.21 1.07  
$ cpp            : num  2.41 3.25 1.36  
$ ctr            : num  1.16 1.071 0.439
```

Response:

```
Failed to connect to 2a03:2880:20:4f06:face:b00c:0:1: Network is unreachable
```

Query sent to Facebook:

```
mystats <- fb_insights(date_preset = 'today', level = 'ad')
```

Query sent to Facebook:

```
mystats <- fb_insights(date_preset = 'today', level = 'ad')
```

Response:

```
Curl (52): Empty reply from server
```


Query sent to Facebook:

```
mystats <- fb_insights(date_preset = 'today', level = 'ad')
```

Query sent to Facebook:

```
mystats <- fb_insights(date_preset = 'today', level = 'ad')
```

Response header:

```
{  
  "Vary": ["Accept-Encoding"],  
  "Content-Type": ["text/html"],  
  "X-FB-Debug": ["..."],  
  "Date": ["Thu, 24 Sep 2015 16:38:27 GMT"],  
  "Connection": ["keep-alive"],  
  "Content-Length": ["19"],  
  "status": ["503"],  
  "statusMessage": ["Service Unavailable"]  
}
```

Query sent to Facebook:

```
mystats <- fb_insights(date_preset = 'today', level = 'ad')
```

Query sent to Facebook:

```
mystats <- fb_insights(date_preset = 'today', level = 'ad')
```

Header:

```
{"Content-Type":["text/html; charset=utf-8"], ..., "status":["502"],  
"statusMessage":["Error parsing server response"]}
```

Query sent to Facebook:

```
mystats <- fb_insights(date_preset = 'today', level = 'ad')
```

Header:

```
{"Content-Type":["text/html; charset=utf-8"], ..., "status":["502"],  
"statusMessage":["Error parsing server response"]}
```

Response:

```
<!DOCTYPE html>  
<html lang="en" id="facebook">  
  <head>  
    <title>Facebook | Error</title>  
  </head>  
  <body>  
    <h1 id="sorry">Sorry, something went wrong.</h1>  
    <p id="promise">We're working on it and we'll get it fixed as soon as we can.</p>  
  </body>  
</html>
```

Query sent to Facebook:

```
mystats <- fb_insights(date_preset = 'today', level = 'ad')
```

Query sent to Facebook:

```
mystats <- fb_insights(date_preset = 'today', level = 'ad')
```

Header:

```
{"Content-Type":["application/json; charset=UTF-8"], ..., "status":["200"],  
"statusMessage":["OK"]}
```

Query sent to Facebook:

```
mystats <- fb_insights(date_preset = 'today', level = 'ad')
```

Header:

```
{"Content-Type":["application/json; charset=UTF-8"], ..., "status":["200"],  
"statusMessage":["OK"]}
```

Response:

```
{  
  "id":["..."],  
  "account_id":["..."],  
  "time_ref":[...],  
  "async_status":["Job Failed"],  
  "async_percent_completion":[0]  
}
```


Possible issues with the API calls:

- 1 Network error (network is unreachable)
- 2 Curl error (52)
- 3 HTTP error (503)
- 4 JSON syntax error (HMTL)
- 5 Facebook API error message

Possible issues with the API calls using `fbRads` from `R` :

- 1 ~~Network error (network is unreachable)~~
- 2 ~~Curl error (52)~~
- 3 ~~HTTP error (503)~~
- 4 ~~JSON syntax error (HMTL)~~
- 5 ~~Facebook API error message~~

```
> mystats <- fb_insights(date_preset = 'today', level = 'ad')
```

```
ERROR [2015-11-01 08:27:44] Possible network error: Empty reply from server
INFO [2015-11-01 08:28:14] Retrying query for the 1 st/nd/rd time
ERROR [2015-11-01 08:28:14] Possible network error: Empty reply from server
INFO [2015-11-01 08:28:44] Retrying query for the 2 st/nd/rd time
DEBUG [2015-11-01 08:28:44] Sync request failed, starting async request.
DEBUG [2015-11-01 08:28:45] *** Async Job Not Started (0%). Waiting 2 seconds...
DEBUG [2015-11-01 08:28:47] *** Async Job Started (0%). Waiting 10 seconds...
ERROR [2015-11-01 08:28:57] {"id":["***"],..., "async_status":["Job Failed"]}
INFO [2015-11-01 08:28:57] Retrying query for the 1 st/nd/rd time
DEBUG [2015-11-01 08:28:57] *** Async Job Not Started (0%). Waiting 2 seconds...
DEBUG [2015-11-01 08:29:00] *** Async Job Started (0%). Waiting 10 seconds...
DEBUG [2015-11-01 08:29:10] *** Async Job Running (17%). Waiting 7.5 seconds...
DEBUG [2015-11-01 08:29:17] *** Async Job Running (35%). Waiting 5.6 seconds...
DEBUG [2015-11-01 08:29:23] *** Async Job Running (53%). Waiting 4.2 seconds...
DEBUG [2015-11-01 08:29:28] *** Async Job Running (71%). Waiting 3.2 seconds...
DEBUG [2015-11-01 08:29:31] *** Async Job Running (71%). Waiting 15.8 seconds...
```

```
> mystats <- fb_insights(date_preset = 'today', level = 'ad')
```

```
DEBUG [2015-11-01 08:28:56] Sync request failed, starting async request.
DEBUG [2015-11-01 08:28:57] *** Async Job Not Started (0%). Waiting 2 seconds...
DEBUG [2015-11-01 08:29:00] *** Async Job Started (0%). Waiting 10 seconds...
DEBUG [2015-11-01 08:29:10] *** Async Job Running (17%). Waiting 7.5 seconds...
DEBUG [2015-11-01 08:29:17] *** Async Job Running (35%). Waiting 5.6 seconds...
DEBUG [2015-11-01 08:29:23] *** Async Job Running (53%). Waiting 4.2 seconds...
DEBUG [2015-11-01 08:29:28] *** Async Job Running (71%). Waiting 3.2 seconds...
DEBUG [2015-11-01 08:29:31] *** Async Job Running (71%). Waiting 15.8 seconds...
```

<https://github.com/cardcorp/fbRads>



```
## Number of R users on Facebook
> (fbr <- fb_ad_get_search(q = 'rstats', type = 'adinterest')[, c(1:3, 6)])
      id                name audience_size      topic
1 6003212345926 R (programming language)    1602320 Lifestyle and culture
```

```
## Number of R users on Facebook
```

```
> (fbr <- fb_ad_get_search(q = 'rstats', type = 'adinterest')[, c(1:3, 6)])  
      id                name audience_size      topic  
1 6003212345926 R (programming language)    1602320 Lifestyle and culture
```

```
## Number of programmers on Facebook
```

```
> fbprog <- fb_ad_get_search(q = 'programming language', type = 'adinterest')  
> head(fbprog[order(fbprog$audience_size, decreasing = TRUE), ], 10)[, 1:3]  
      id                name audience_size  
1  6003030200185      Programming language    269482400  
67 6003017204650      PHP                    37701920  
2  6003476678525      Boo (programming language)    31028180  
68 6004131486306      C++                    26812460  
69 6003215894612      Ajax (programming)          14547070  
3  6003682002118 Python (programming language)    14286850  
70 6003127967124      JavaScript                12124380  
4  6002979703120      Ruby (programming language)    11146690  
5  6003437022731      Java (programming language)    9547610  
71 6003568029103      Object-oriented programming    9490910
```

```
## US-based R users on Facebook
> fb_ad_reach_estimate(targeting_spec = list(
+   geo_locations = list(countries = 'US'),
+   flexible_spec = list(list(
+     interests = data.frame(
+       id = fbr$id,
+       name = fbr$name))))))$users
[1] 200000
```



```
## US-based R users on Facebook
> fbad_reachestimate(targeting_spec = list(
+   geo_locations = list(countries = 'US'),
+   flexible_spec = list(list(
+     interests = data.frame(
+       id = fbr$id,
+       name = fbr$name))))))$users
[1] 200000
```

```
> fbprog <- data.table(fbprog)[name %in% c(
+   'R (programming language)',
+   'Python (programming language)',
+   'Java (programming language)']
```

```
## US-based R, Python or Java users on Facebook
> fbad_reachestimate(targeting_spec = list(
+   geo_locations = list(countries = 'US'),
+   flexible_spec = list(list(
+     interests = data.frame(
+       id = fbprog$id,
+       name = fbprog$name))))))$users
[1] 1700000
```

```
## US-based R, but non-Python or Java users on Facebook
> fbad_reachestimate(targeting_spec = list(
+   geo_locations = list(countries = 'US'),
+   flexible_spec = list(list(
+     interests = data.frame(id = fbr$id, name = fbr$name))),
+   exclusions = list(interests = data.frame(
+     id = fbprog$id[1:2],
+     name = fbprog$name[1:2]))))$users
[1] 190000
```

```
## US-based R, Python and Java users on Facebook
> fbad_reachestimate(targeting_spec = list(
+   geo_locations = list(countries = 'US'),
+   flexible_spec = list(
+     list(interests = data.frame(
+       id = fbprog$id[1],
+       name = fbprog$id[1])),
+     list(interests = data.frame(
+       id = fbprog$id[2],
+       name = fbprog$id[2])),
+     list(interests = data.frame(
+       id = fbprog$id[3],
+       name = fbprog$id[3]))))$users
[1] 5300
```

