
12:30
EARL
EFFECTIVE APPLICATIONS OF THE R LANGUAGE
BOSTON
2-4 NOVEMBER
2015
How to create ads from R?
www.earl-conference.com
Learn how to create Facebook ads from R at
EARL 2015 in Boston on Nov 3 2015.

fbRads

Analyzing and managing Facebook ads from R

Ajay Gopal, Gergely Daroczi

LA-R meetup

November 17 2015

About us

CARD.COM

- Age: 3 yr 1 mo
- Strength: ~50
- Location: Santa Monica, CA

$20M Raised Till Date, over 500K Cardholders & **Growing Fast**!

Ajay [@aj2z] VP, Growth & Data Science

Gergely [@daroczig] Lead R Developer

Next 45 mins

CARD.COM

- How & why fbRads was born
- Facebook ad basics
- Detailed package demo

Our mission

CARD.COM

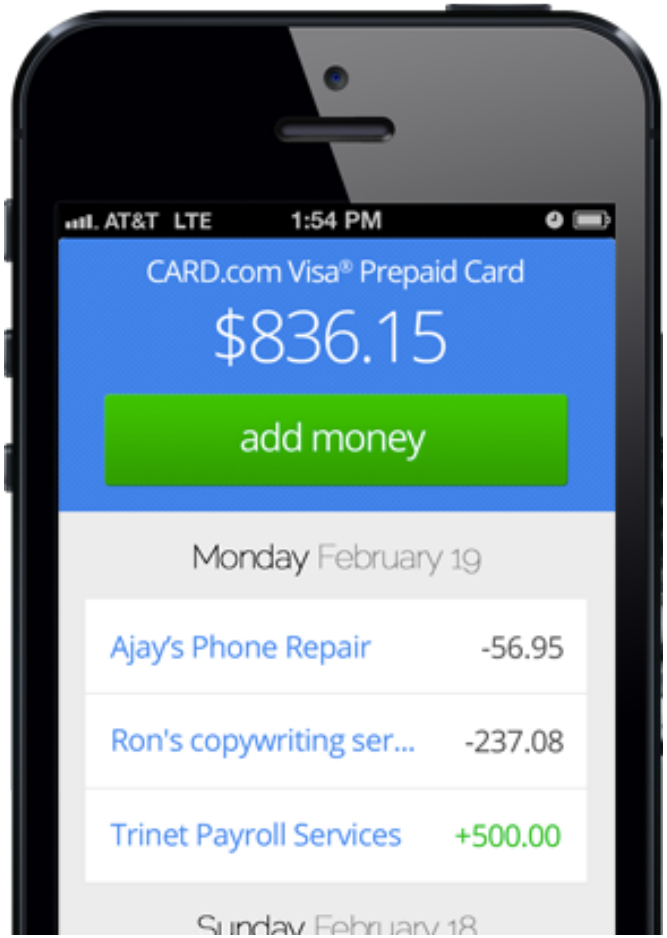
Personalized Financial Service

- Alternative to B&M Banking
- Mobile
- Affordable
- Fair
- Accessible to all

. . . and also fun and fashionable

CARD.COM

Mobile App



General Purpose Reloadable Prepaid Card Customized with whatever inspires you!

Our Rewards: Inspi3ation in your wallet!

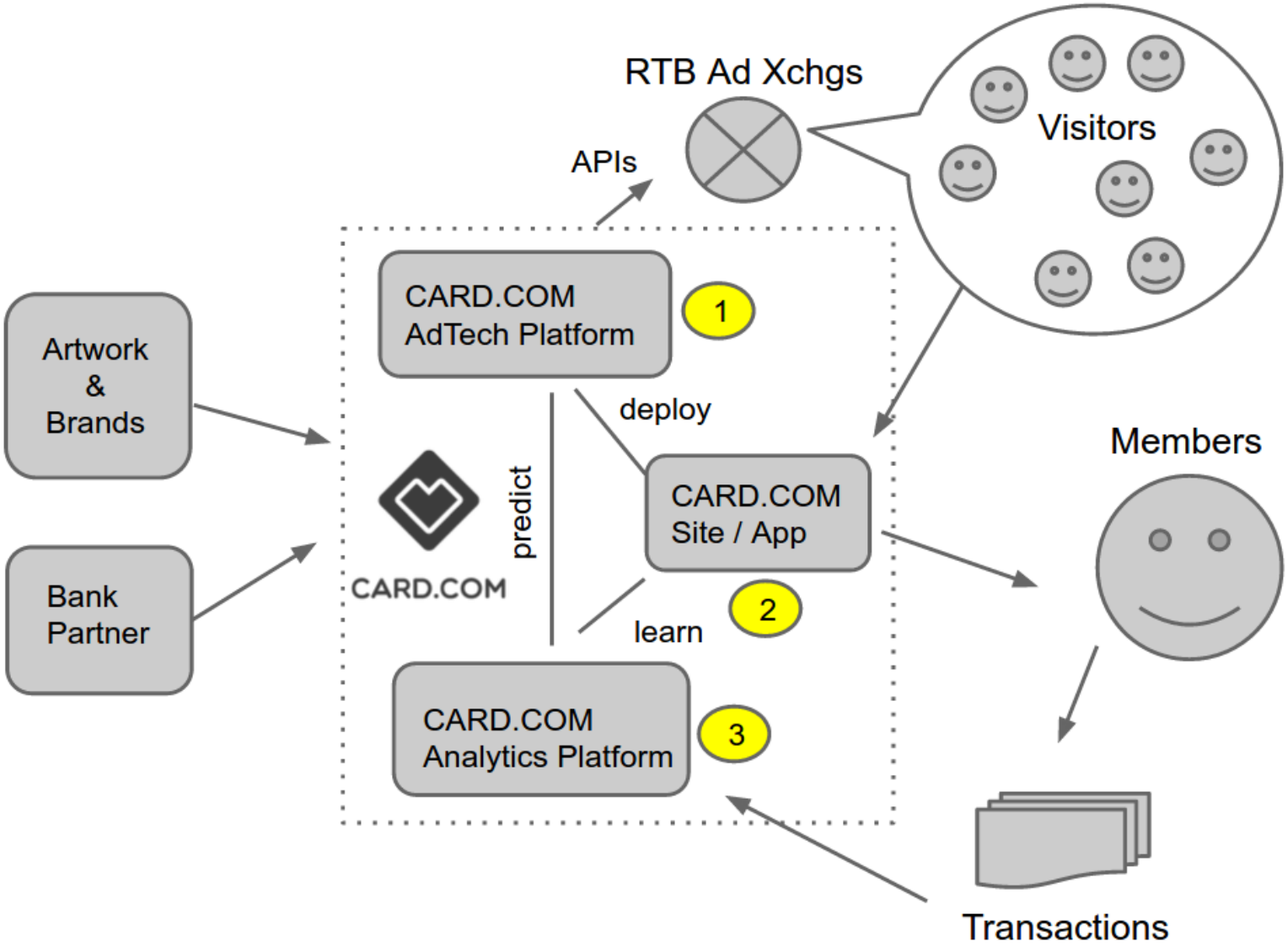
Scale diverse ads with 3 employees

CARD.COM

≠ ≠

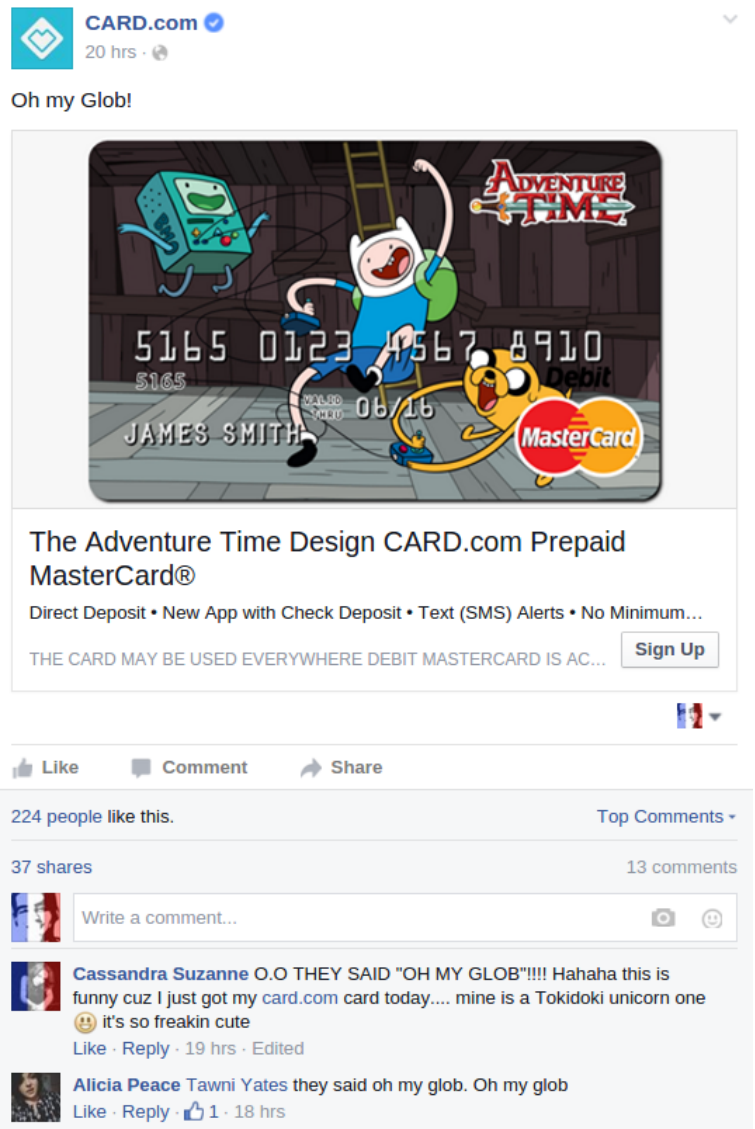
≠

Soln: Modern Marketing



Where to advertise?

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



Ad Platforms

CARD.COM

- Utilize Google search data via AdWords API

```
## by Johannes Burkhardt
devtools::install.github('jburkhardt/RAdwords')
```

- Utilize Amazon purchase history via Amazon Ads

```
NULL
```

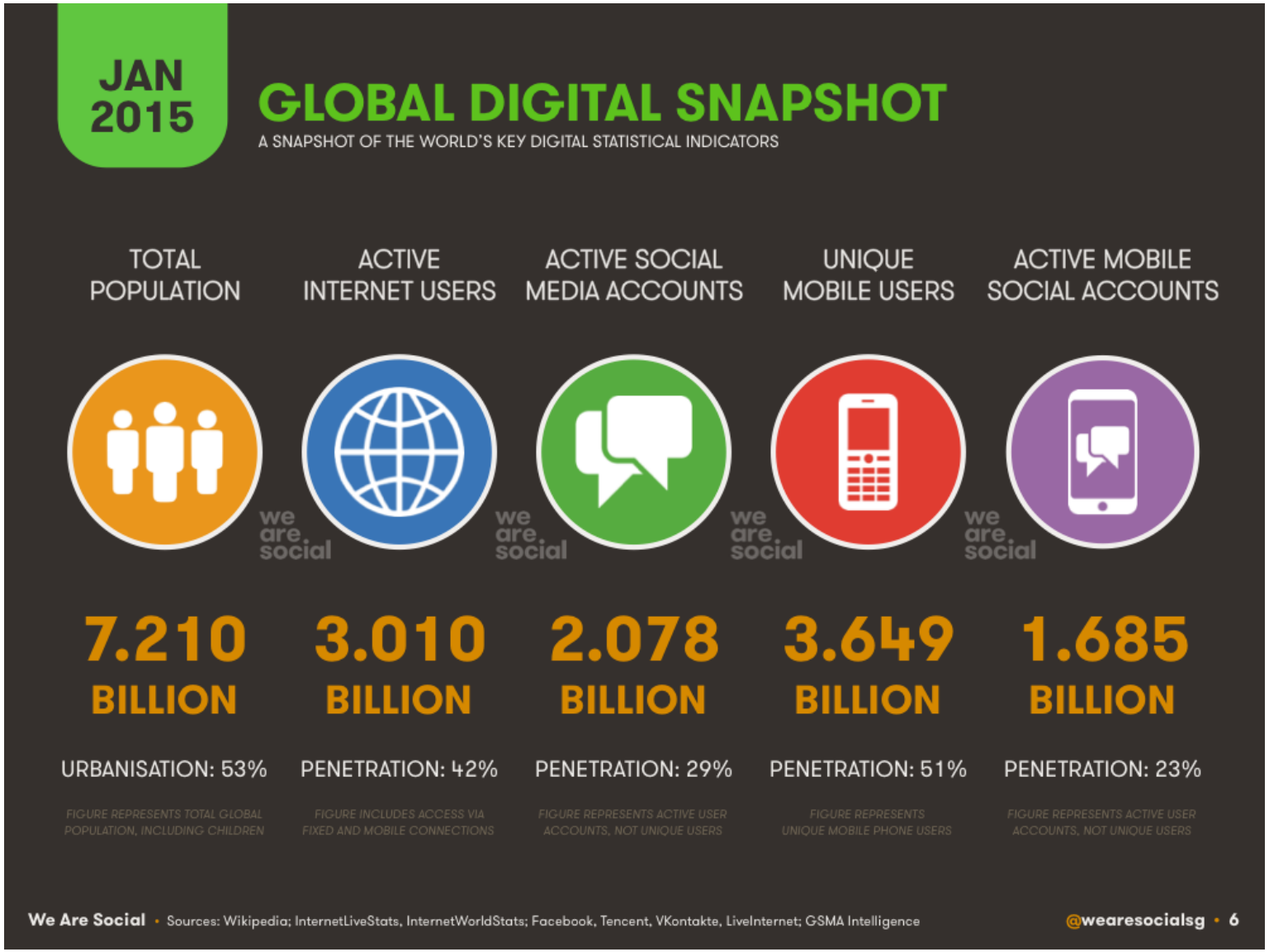
- Yahoo+Bing have joint search ad network & API

```
NULL
```

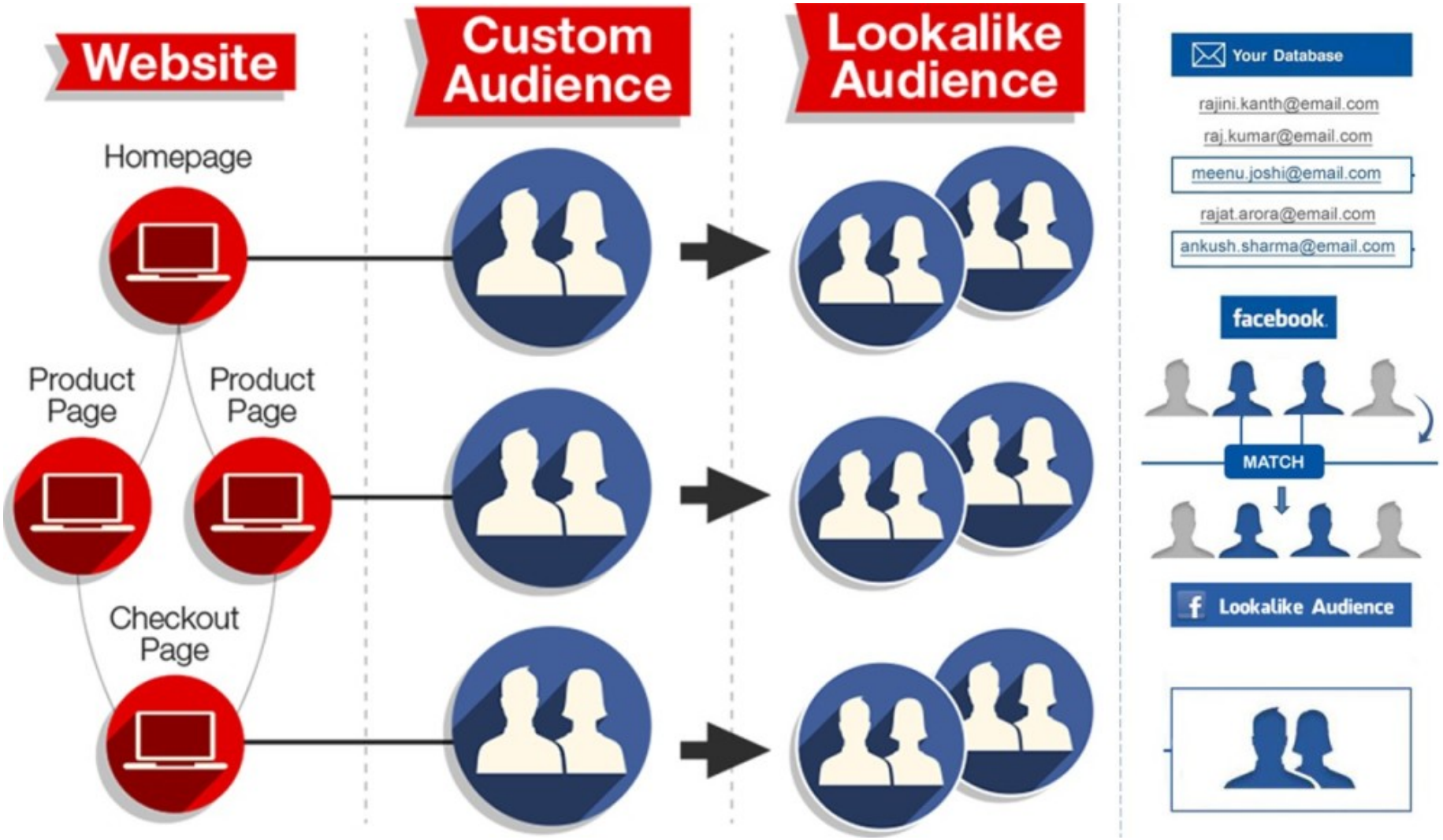
- Utilize Facebook likes & comments data via FB Marketing API

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

Advertising on Facebook

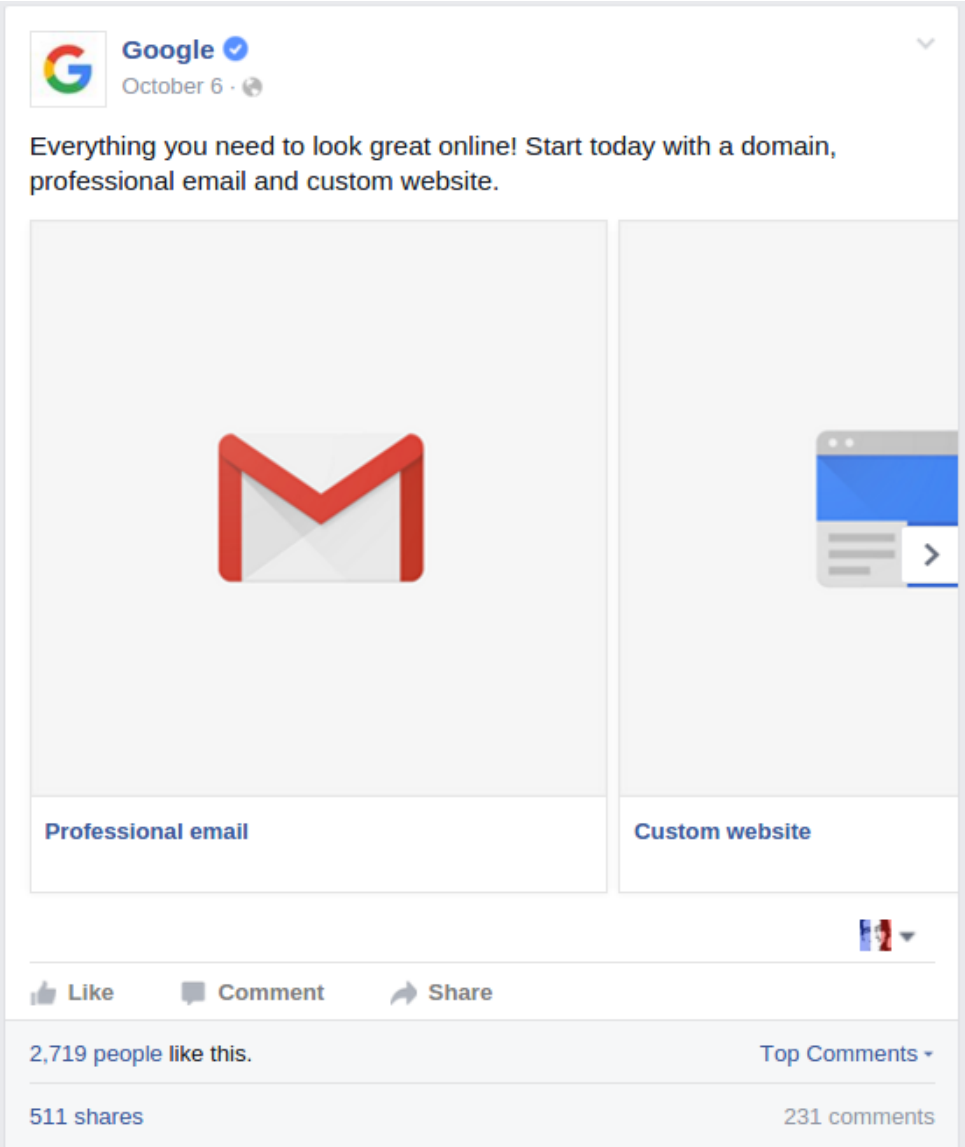


Advertising on Facebook

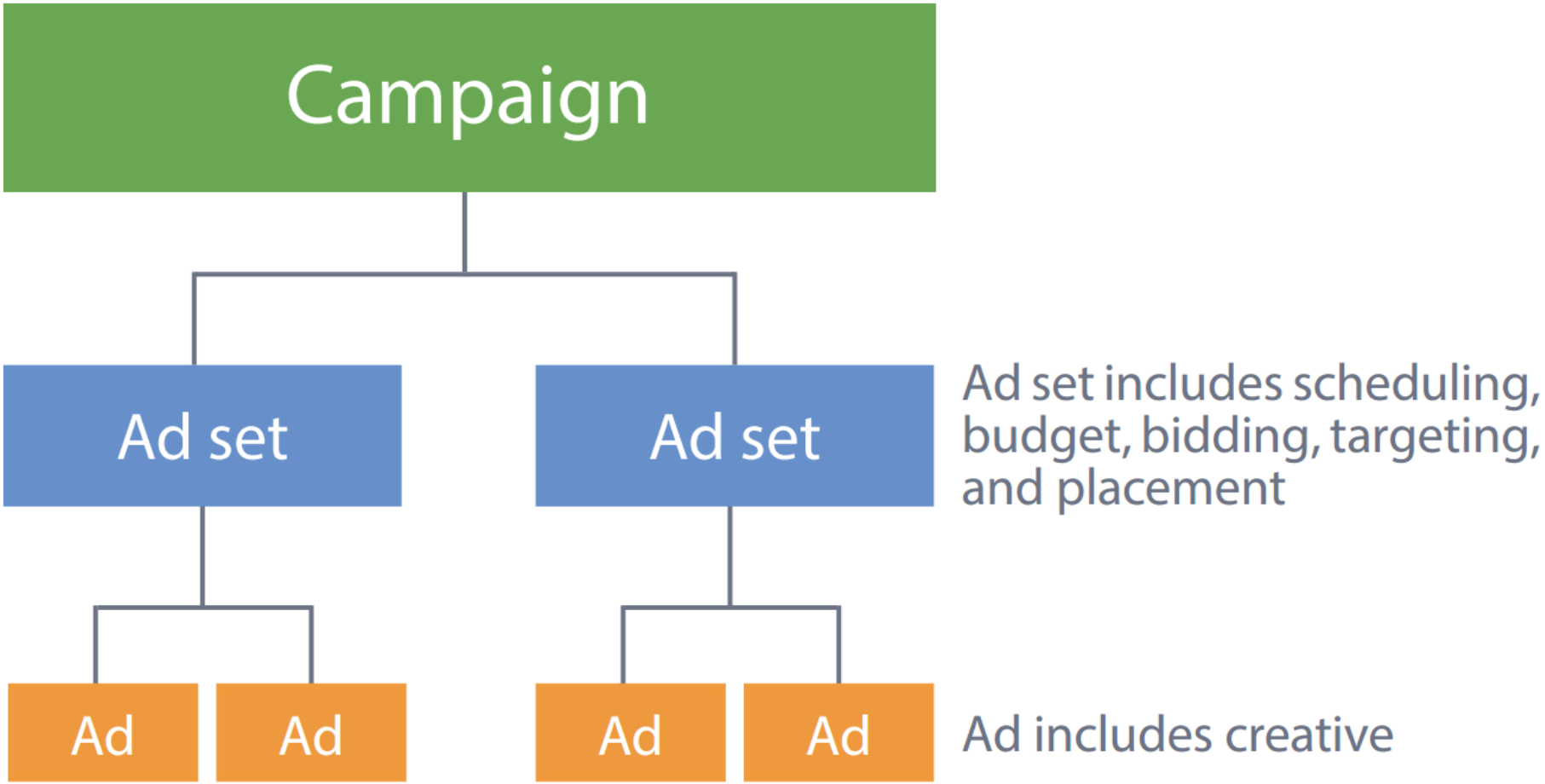


Source: adparlor.com

Who advertises on Facebook



Anatomy of a Facebook Ad Campaign



Source: Facebook Marketing API docs

Facebook Ad Campaign Changes

- Anyone here from Facebook?

Facebook Ad Campaign Changes

CARD.COM

- Anyone here from Facebook?

- 2 changes in the campaign structure in 2014
- 2 -> 3 hierarchical categories
- Before July 2014, "Ad Sets" were called "Campaigns"
- At the API endpoints:
  - campaigns are called `adcampaign_groups`
  - ad sets are called `adcampaigns`
  - ads are called `adgroups`
- When creating an ad via the API, the adset id is called `campaign_id`
- 4 new Facebook Marketing API versions in October 2014
- new API version every 6 months

Facebook Ad Campaign Changes

CARD.COM

- Anyone here from Facebook?

- 2 changes in the campaign structure in 2014
- 2 -> 3 hierarchical categories
- Before July 2014, "Ad Sets" were called "Campaigns"
- At the API endpoints:
  - campaigns are called `adcampaign_groups`
  - ad sets are called `adcampaigns`
  - ads are called `adgroups`
- When creating an ad via the API, the adset id is called `campaign_id`
- 4 new Facebook Marketing API versions in October 2014
- new API version every 6 months

- But it's pretty damn good

Facebook Ad Campaign Changes

CARD.COM

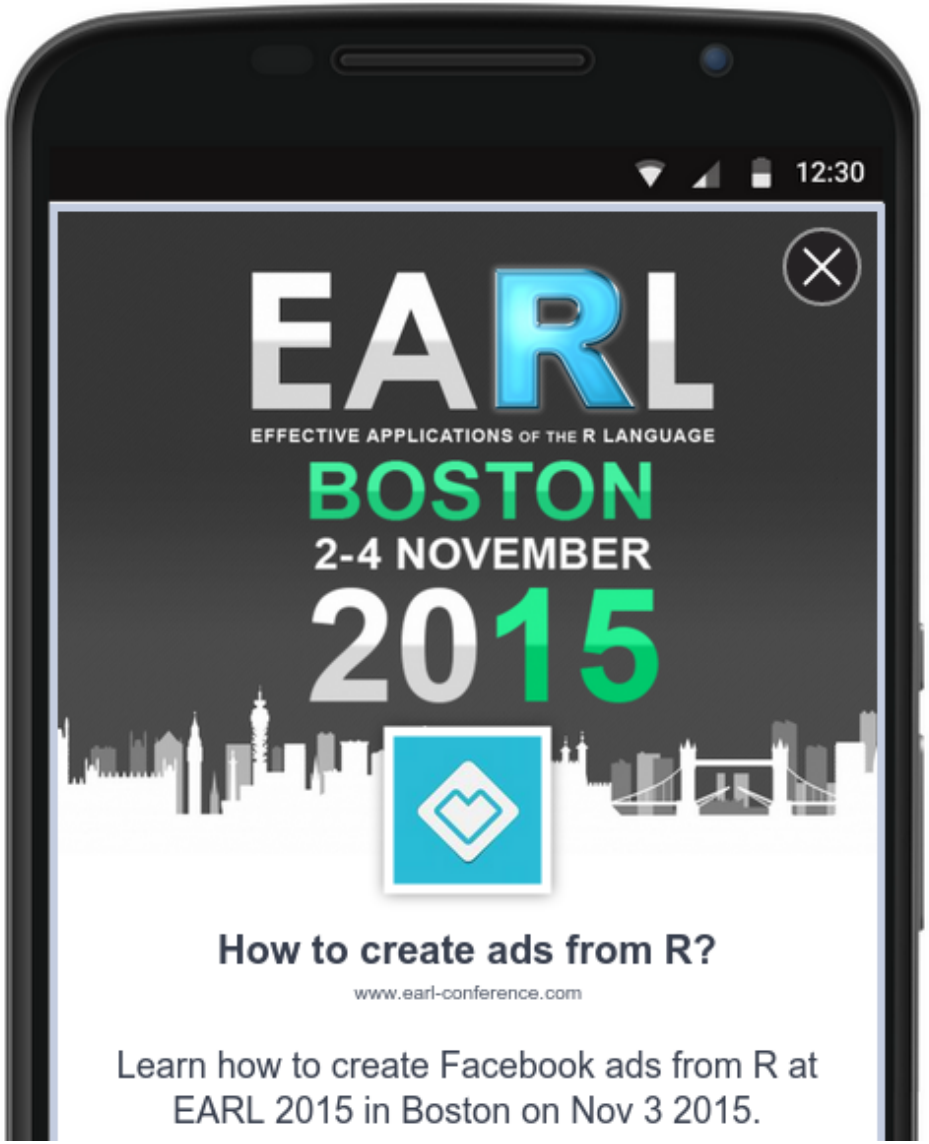
- Anyone here from Facebook?

- 2 changes in the campaign structure in 2014
- 2 -> 3 hierarchical categories
- Before July 2014, "Ad Sets" were called "Campaigns"
- At the API endpoints:
  - campaigns are called `adcampaign_groups`
  - ad sets are called `adcampaigns`
  - ads are called `adgroups`
- When creating an ad via the API, the adset id is called `campaign_id`
- 4 new Facebook Marketing API versions in October 2014
- new API version every 6 months

- But it's pretty damn good

- Really!

Has anyone seen this?



Has anyone seen this?

CARD.COM

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-mail addres

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

Get R package developer e-mail addres

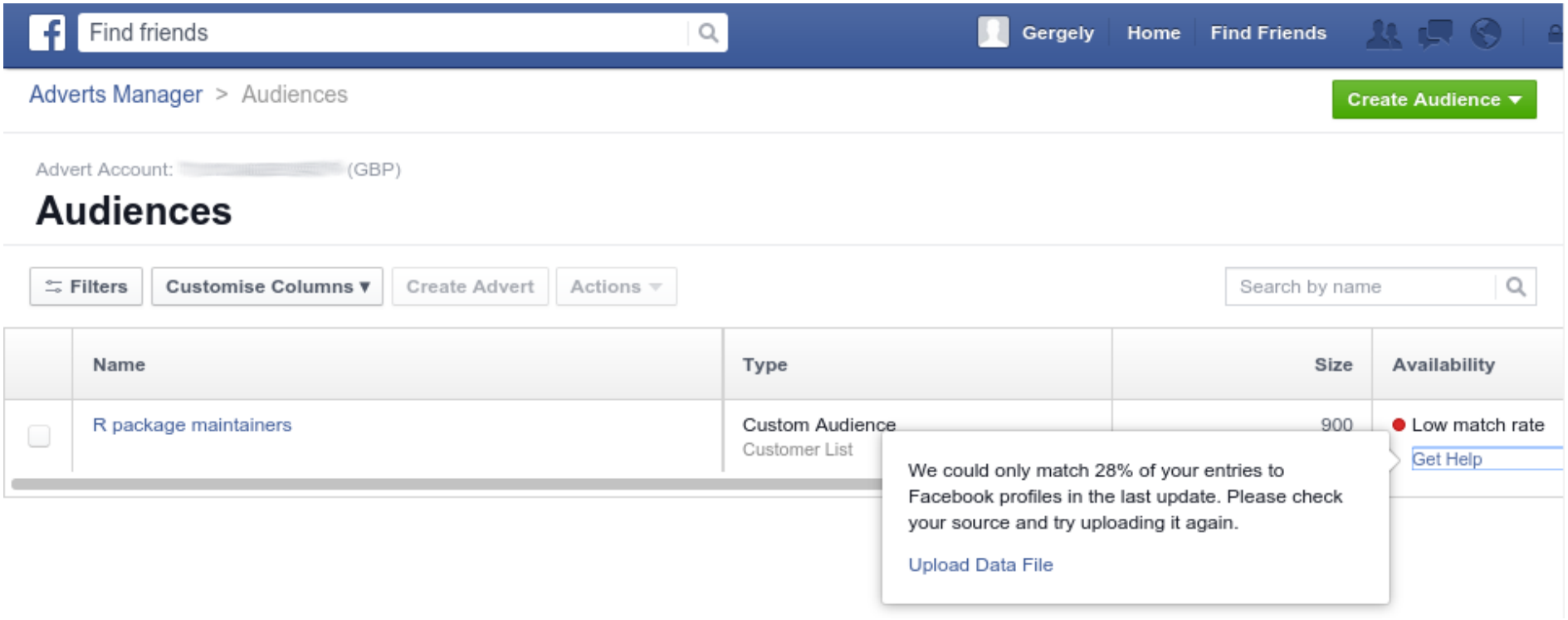
```
> url <- 'http://cran.r-project.org/web/checks/check_summary.html'
> packages <- readHTMLTable(url, which = 2)
> mails <- sub('.*<(.*)>', '\\1', packages$' Maintainer')
> mails <- sub(' at ', '@', mails)
```

```
> tail(sort(table(mails)))
## Dirk Eddelbuettel (35)
## Kurt Hornik (29)
## Scott Chamberlain (24)
## Martin Maechler (24)
## Paul Gilbert (22)

> length(unique(mails))
## 4023

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

Get R package developer e-mail addres



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

Get e-mails from [R-help]

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')
```

Extract e-mail addresses from [R-help]

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)
```

Verify e-mail addresses from [R-help]

```
> 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
```

Verify e-mail addresses from [R-help]

```
> 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.
```

Verify e-mail addresses from [R-help]

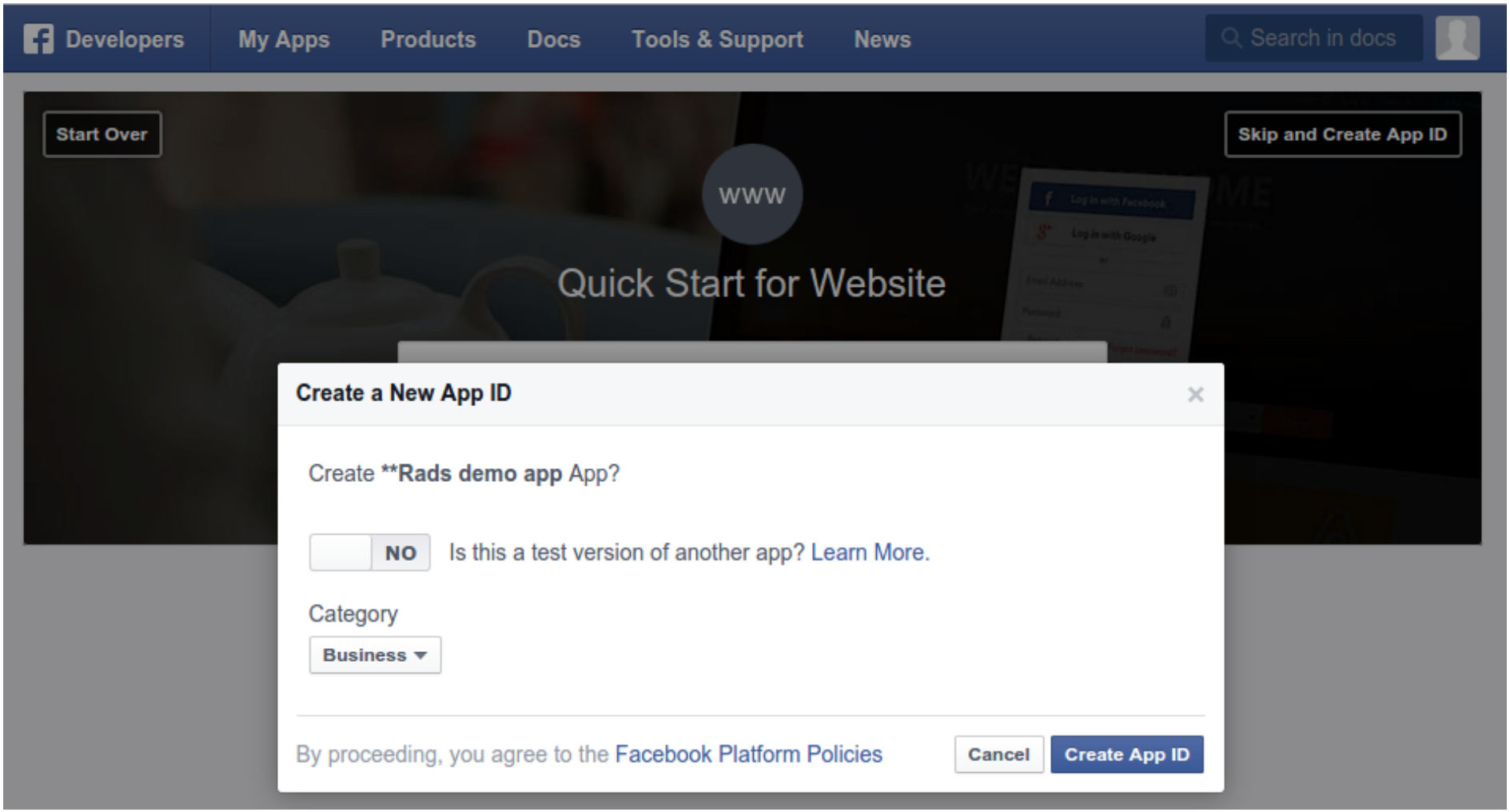
```
> length(mails)
266449

> 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/
```

Authenticate with the Facebook API



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

Authenticate with the Facebook API

CARD.COM

Create a token:

```
> library(httr)
> app <- oauth_app('facebook', 'your_app_id', 'your_app_secret')
> 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')
```

The fun begins!

Initialize connection to Facebook Marketing API:

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

Create custom audience

```
> 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
```

Create lookalike audiences

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)
+
+ })
```

Read lookalike audiences

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
```

Create a campaign

```
> campaign <- fbad_create_campaign(
+     name = 'Promote my EARL 2015 Boston talk')
> fbad_read_campaign(id = campaign)
$id
[1] "********"

$account_id
[1] "********"

$buying_type
[1] "AUCTION"

$campaign_group_status
[1] "ACTIVE"

$objective
[1] "NONE"

$name
[1] "Promote my EARL 2015 Boston talk"
```

Define target for an adset

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)
```

Create an adset

This is where we define the target and budget:

```
> adset <- fbad_create_adset(
+     name = 'Promo budget for my EARL 2015 Boston talk',
+     campaign_group_id = campaign,
+     billing_events = 'IMPRESSIONS',
+     optimization_goal = 'REACH',
+     bid_amount = 5,
+     campaign_status = 'ACTIVE',
+     lifetime_budget = 7000,
+     end_time = as.numeric(as.POSIXct('2015-11-03')),
+     targeting = target)
```

Upload image

Get an image for the ad:

```
> img <- 'EARL-2015-Boston.png'
> download.file('http://www.earl-conference.com/boston/Images/Header.jpg', img)
```

Upload to Facebook:

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

Take a note on the returned hash:

```
> str(img)
List of 3
 $ filename: chr "EARL-2015-Boston.png"
 $ hash    : chr "7607044f3a90533f7aa3cd98d3b08ee0"
 $ url     : chr "https://scontent.xx.fbcdn.net/hads-xta1/t45.1600-4/121248
> img <- img$hash
```

Create a creative

```
> url <- paste0('http://www.earl-conference.com/boston/',
+               'speakers/speaker.php?s=gergely_daroczi'),
> creative <- fbad_create_creative(
+     name = 'How to create ads from R?',
+     body = paste(
+         'Learn how to create Facebook ads from R',
+         'at EARL 2015 in Boston',
+         'on Nov 3 2015.'),
+     title      = 'How to create ads from R?',
+     object_url = url,
+     image_hash = img$hash)
```

Preview the creative

Create an ad

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

Scalability

A/B testing

```
> taglines <- c('How to manage ads from R?',
+               'How to optimize ads from R?')

> for (tagline in taglines) {
+
+     ## create creative
+     creative <- fbad_create_creative(
+         name = tagline,
+         body = paste(
+             'Learn how to create Facebook ads from R',
+             'at EARL 2015 in Boston',
+             'on Nov 3 2015.'),
+         title      = tagline,
+         object_url = url,
+         image_hash = img$hash)
+
+     ## create ad
+     ad <- fbad_create_ad(
+         name        = paste0(tagline),
+         campaign_id = adset,
+         creative    = creative)
+
+ }
```

Performance metrics

CARD.COM

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 [redacted] ▸ Advert Set: My budget for [redacted]

How does this page work? | Create Advert in Advert Set

STATUS	DELIVERY	SPENT TODAY	LIFETIME SPENT	END DATE
	Active	[redacted]	[redacted]	[redacted]

RESULTS ?	REACH ?	FREQUENCY ?	TOTAL SPENT ?	COST PER RESULT ?
—	[redacted]	[redacted]	[redacted]	—

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 ?
	[redacted]	Active	Not Available	—	[redacted]	[redacted]	[redacted]	[redacted]	7/10	[redacted]	[redacted]	[redacted] CPC	[redacted] CPC
	[redacted]	Active	Not Available	—	[redacted]	[redacted]	[redacted]	[redacted]	7/10	[redacted]	[redacted]	[redacted] CPC	[redacted] CPC
	[redacted]	Active	Not Available	—	[redacted]	[redacted]	[redacted]	[redacted]	8/10	[redacted]	[redacted]	[redacted] CPC	[redacted] CPC

3 Results

About | Create Advert | Create Page | Developers | Careers | Privacy | Cookies | AdChoices | Terms | Help

Facebook © 2015
English (UK)

Performance metrics

```
> 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
```

Performance metrics

```
> 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
```

Performance metrics – placeholder slide

```
> 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
```

What Could Possibly Go Wrong?

Query sent to Facebook:

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

What Could Possibly Go Wrong?

CARD.COM

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
```

What Could Possibly Go Wrong?

CARD.COM

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:

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

What Could Possibly Go Wrong?

CARD.COM

Query sent to Facebook:

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

What Could Possibly Go Wrong?

CARD.COM

Query sent to Facebook:

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

Response:

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

What Could Possibly Go Wrong?

Query sent to Facebook:

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

What Could Possibly Go Wrong?

CARD.COM

Query sent to Facebook:

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

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"]
}
```

What Could Possibly Go Wrong?

Query sent to Facebook:

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

What Could Possibly Go Wrong?

CARD.COM

Query sent to Facebook:

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

Header:

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

What Could Possibly Go Wrong?

CARD.COM

Query sent to Facebook:

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

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.</
  </body>
</html>
```

What Could Possibly Go Wrong?

Query sent to Facebook:

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

What Could Possibly Go Wrong?

CARD.COM

Query sent to Facebook:

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

Header:


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

What Could Possibly Go Wrong?

CARD.COM

Query sent to Facebook:

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

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]
}
```

What Could Possibly Go Wrong?

CARD.COM

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

Error handling

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~~

Error handling

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

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...
```

Support for async/batch queries

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

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

Keyword-based targeting

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

Keyword-based targeting

```
## Number of R users on Facebook
> (fbr <- fbad_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 <- fbad_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
```

Targeting overlaps

```
## 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
```

Targeting overlaps

```
## 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
```

Targeting overlaps

```
## 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
```

Targeting overlaps