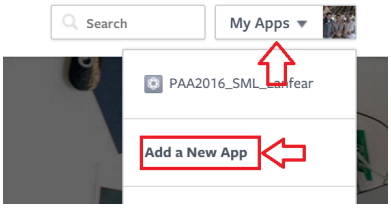
Facebook with Rfacebook and SocialMediaLab

Charles Lanfear, University of Washington

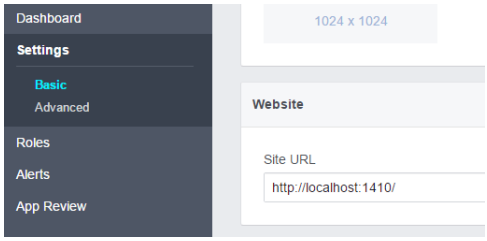
October 10, 2016

Facebook Setup

To obtain data using Facebook's API, we need to create an app on Facebook's developer website (https://developers.facebook.com). Follow the link to https://developers.facebook.com and log in using your Facebook account using the button in the top right of the window. Once logged in, use the drop down menu in the same location to select "Add a New App":



Then, select "Website" and give your app any name–such as "ALAP2016_YourName"–and assign it any category. Then click "Skip and Create App ID" in the top right to continue. You will then be prompted to create a display name for the app (you can choose anything you wish), assign an email address, and choose a category (such as education). You will not be presented with the developer dashboard. Go to Settings on the left and then at the bottom of the basic settings pane, enter http://localhost:1410/ into the "Site URL" field as you see below. If this field is not displayed, click on "Add Platform" and select "Website" to add the field.



At the top of the dashboard you will find app information for authentication with the Facebook API:

You will need to copy your "App ID" and "App Secret" (press "Show" to obtain it) into specific locations in the next part of this tutorial to authenticate. Do not use the example App ID in the image above and do not share your App Secret code with anyone else. For more information on your App Secret code, click on this link. With your App ID and App Secret, you should now be able to authorize packages like Rfacebook and SocialMediaLab to access the Facebook API through your account. These codes are not limited to use with R packages and can be used with any software built to utilize the Facebook API.

---

2) R Setup

Before we can started, we need to clear our workspace, set a working directory, and install and load packages we'll be using. You will need to fill in your working directory below by replacing "FILL IN HERE" with the location on your computer where the workshop folder was placed.

```
rm(list=ls())
setwd("FILL IN HERE/ALAP_2016_Workshop/Facebook")

install.packages("SocialMediaLab")
install.packages("Rfacebook")
install.packages("plyr")
install.packages("stringr")
install.packages("dplyr")
install.packages("igraph")

library(SocialMediaLab)
library(Rfacebook)
library(plyr)
library(stringr)
library(dplyr, pos=99) # dplyr and igraph in high position to avoid masking plyr.
library(igraph, pos=100)
```

---

3) Collecting Data with Rfacebook

Authentication

First we need to obtain a token to authorize our access to the Facebook API. For this you will need to fill in your Application ID and Application Secret ID below. These are taken as arguments in SocialMediaLab's authentication function which returns an OAuth token to permit it, and other Facebook packages such as Rfacebook, to access the Facebook API to collect data and perform tasks. The option "extended_permissions" enables additional permissions needed for advanced functions outside the bounds of this workshop and "useCachedToken" tells the function to search your workspace for an existing token rather than creating a new one every time the script is run. If you receive an error saying "Error validating access token: Session has expired..." you should change this to "useCachedToken = FALSE".

```
fb.appid     <- "APPID GOES HERE"
fb.appsecret <- "APP SECRET GOES HERE"
fb_oauth <- AuthenticateWithFacebookAPI(appID = fb.appid,
                                        appSecret = fb.appsecret,
                                        extended_permissions = FALSE, # Public Info
                                        useCachedToken = TRUE)        # Use existing
```

Collecting Posts from Public Pages

The first example of Facebook data collection we will explore is collecting posts from public Facebook pages. Unlike collecting user data, this is not limited to your Facebook network. We will use Rfacebook's getpage() function to get posts from the International Union for the Scientific Study of Population's Facebook page. Comments in the code block below describe the content of the function's arguments.

```
iussp.page.df <- getPage(page="IUSSP",  # Takes page ID or page name
                         token=fb_oauth,   # OAuth token
                         n=25,             # Number of posts to return
                         since=NULL,       # Date of earliest posts returned
                         until=NULL,       # Date of latest posts returned
                         feed=TRUE)        # T/F: Return posts by page non-owners
```

```
## 25 posts
```

We've collected up to 50 posts to the page. We can see that the data this function returns have a number of values, and we can take a look at what they contain:

```
names(iussp.page.df) # Get names of returned columns
```

```
##  [1] "from_id"        "from_name"      "message"        "created_time"
##  [5] "type"           "link"           "id"             "likes_count"
##  [9] "comments_count" "shares_count"
```

```
glimpse(iussp.page.df) # Take a peek at the data
```

```
## Observations: 25
## Variables: 10
## $ from_id        <chr> "769441549840182", "769441549840182", "76944154...
## $ from_name      <chr> "International Union for the Scientific Study o...
## $ message        <chr> "N-IUSSP Article\nThe shrinking population of E...
## $ created_time   <chr> "2016-10-10T07:54:42+0000", "2016-10-03T09:05:5...
## $ type           <chr> "photo", "photo", "status", "link", "link", "st...
## $ link           <chr> "https://www.facebook.com/iussp/photos/a.791490...
## $ id             <chr> "769441549840182_1061778673939800", "7694415498...
## $ likes_count    <dbl> 4, 1, 2, 2, 2, 3, 3, 0, 0, 2, 2, 4, 4, 13, 3, 1...
## $ comments_count <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,...
## $ shares_count   <dbl> 1, 0, 0, 1, 1, 1, 0, 0, 3, 1, 1, 1, 0, 2, 0, 0,...
```

The "message" component of these data, the text of posts, is likely to be the most useful for most researchas we can use it for text analysis, though there are many potential uses for other components. For this workshop, we will make use of text from posts to do sentiment analysis.

---

4) Sentiment Analysis

We can take the text content from pages to do sentiment analysis like one might do with Twitter data; for a fairly popular example, see this sentiment analysis of Trump tweets. Let's pick some pages likely to have a large number of positive and negative words.

```
trump.page.df    <- getPage("DonaldTrump", fb_oauth, n=10, feed=TRUE)
```

```
## 10 posts
```

```
clinton.page.df  <- getPage("hillaryclinton", fb_oauth, n=10, feed=TRUE)
```

```
## 10 posts
```

Now we need to load a function to do sentiment scoring. This small script uses a simple scoring algorithm: Using dictionaries of positive and negative words it takes a sum of the number of words in a provided sentence matching positive terms and subtracts the number of negative terms and returns this integer.

```
source("./Files/sentiment.r") # Script for analysis. Author: Jeffrey Breen
```

To use this, we need dictionaries of positive and negative words. We've provided some relatively basic dictionaries, but you can easily produce your own dictionaries. You need not be limited to "positive" and "negative" terms either; you could compare any concepts using this same method with some careful planning.

```
pos <- readLines("./Files/opinion_lexicon/positive_words.txt") # Positive words
neg <- readLines("./Files/opinion_lexicon/negative_words.txt") # Negative words

# Let's see what sort of words are "positive" or "negative".
sample(pos, size=9)
```

```
## [1] "poised"       "simplify"     "breathtaking" "enviable"
## [5] "recovery"     "dignify"      "humility"     "equitable"
## [9] "harmless"
```

```
sample(neg, size=9)
```

```
## [1] "aggrieve"       "interference"   "passiveness"    "less-developed"
## [5] "thug"           "betray"         "gimmicks"       "rampage"
## [9] "misleadingly"
```

Now we use our sentiment function and dictionaries to score the text data from the pages.

```
clinton.ss <- score.sentiment(sentence  = clinton.page.df$message, # Text to score
                              pos.words = pos,                     # Positive words
                              neg.words = neg)$score               # Negative words

trump.ss   <- score.sentiment(trump.page.df$message, pos, neg)$score
```

If you get an error about unrecognized characters, you may have characters the sentiment function was not able to filter out. This might require additional data cleaning using regular expressions and does happen occasionally. Learning to use regular expressions is very useful if you plan to work with text data from social media. Here is a good starting point for regular expressions in R: http://www.regular-expressions.info/rlanguage.html

Now let's get some summary statistics on our sentiment scores. First we need to generate a standard error of the mean function, then display our results in a simple data frame.

```
st.err <- function(x){ return( sd(x) / sqrt( length(x) ) ) }

data.frame(Mean=c(mean(clinton.ss), mean(trump.ss)),
           SE=c(st.err(clinton.ss), st.err(trump.ss)),
           row.names=c("Clinton", "Trump"))
```

```
##         Mean        SE
## Clinton  0.3 0.2603417
## Trump    1.1 0.5044249
```

---

5) Searching for Pages

What if have a topic we're interested in collecting page data from, but we don't know what pages we want to use? Rfacebook includes a function for page searches. Here we will search for pages including the word "demography" in their name and see what sort of pages we get and what kind of data this search returns.

```
demography.search.df <- searchPages("Demography", fb_oauth, n=20)
```

```
## 20 pages
```

```
# What sort of values do we get?
names(demography.search.df)
```

```
##  [1] "id"                  "about"               "category"
##  [4] "description"         "general_info"        "likes"
##  [7] "link"                "city"                "state"
## [10] "country"             "latitude"            "longitude"
## [13] "name"                "talking_about_count" "username"
## [16] "website"
```

```
# Let's see what pages we found.
demography.search.df$name
```

```
##  [1] "Demography"
##  [2] "Animal Demography Unit"
##  [3] "DemoGraphy"
##  [4] "Demography (journal)"
##  [5] "Demography of Japan"
##  [6] "Center for Studies in Demography and Ecology"
##  [7] "Demography and Sociology/Demography, U.C. Berkeley"
##  [8] "Demography of the United States"
##  [9] "Medieval demography"
## [10] "Historical demography"
## [11] "Divorce demography"
## [12] "Demography of the United Kingdom"
## [13] "Demography - the study of human population and society"
## [14] "UTSA Department of Demography"
## [15] "DEMOGRAPHY - Institute for Population and Human Studies"
## [16] "Stockholm University Demography Unit - SUDA"
## [17] "Demography (album)"
## [18] "Demography of Birmingham"
## [19] "Demography of Afghanistan"
## [20] "Demography and Population Studies, University of the Witwatersrand"
```

---

6) Basic Network Data with SocialMediaLab

Now that that we've seen it is relatively easy to obtain Facebook page data, we might be interested in doing something other than text analysis. SocialMediaLab provides simple tools for doing network analysis of data from Facebook (as well as other social media platforms). As an example we will download a selection of posts from a public Facebook page and show how posters are connected.

CollectDataFacebook() is used to get network data from Facebook pages, which can be restricted to a specific date range and number of posts.

```
trump.network.df <- CollectDataFacebook(pageName="DonaldTrump",
                                        rangeFrom="2016-10-01",
                                        rangeTo="2016-10-03",
                                        writeToFile=FALSE,
```

```
                                        verbose=TRUE,
                                        n=5)
```

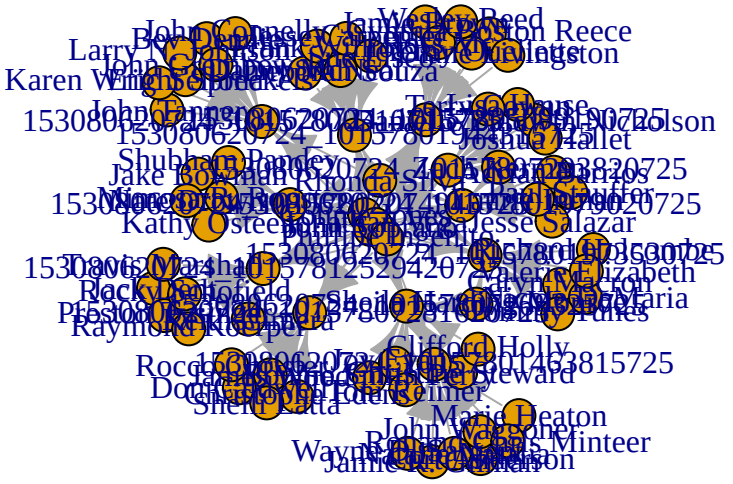
```
## Now retrieving data from page: DonaldTrump
## 2016-10-01  7 posts 2016-10-02  4 posts
## Now collecting data from POSTS (within the page).
##  Collecting posts data from 11 posts.
## Collecting maximum of 5 comments and likes for each post.
## ..........
```

We can then take these network data and use SocialMediaLab and igraph to display network plot. We could also use TclTk to plot an interactive network graph, using the commented out code below.

```
trump.network <- Create(trump.network.df, type="bimodal")
```

```
##
## Creating Facebook bimodal network...
##
## Done!
```

```
plot(trump.network)
```




```
# tkplot(trump.network, canvas.width=800, canvas.height=800) # TclTk alternative
```

Thank you for reading and we hope you found this brief introduction to using R to access the Facebook API. Check out the following links for more information on using Facebook with R.