



IIT School of Applied Technology

ILLINOIS INSTITUTE OF TECHNOLOGY

information technology & management

529 Advanced Data Analytics

November 1, 3 2016

Weekly 11 Presentation

Week 10 Topic: Agenda

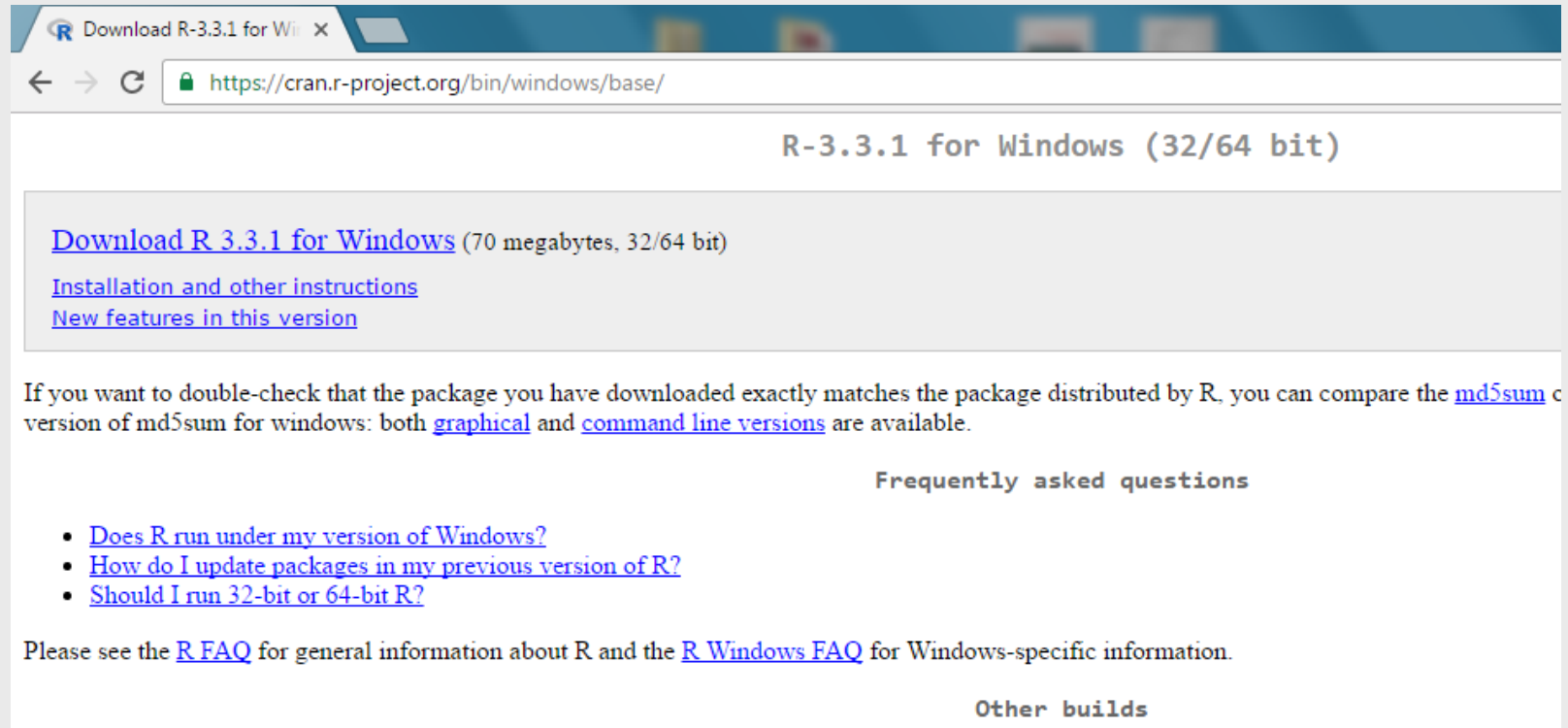
Sentiment Analysis of Tweets:

- ◆ Week 10:
 - Create a Dev account on Twitter
 - Store tweets in corpus/DF
 - Develop own sentiment scoring function
- ◆ Week 11: Visualize analytics in Tableau
- ◆ Week 12: Store tweets in Hadoop
- ◆ Week 13: Store tweets in MongoDB

- ◆ Final Exam update

Week 10 Topic: R Version Needed

- ◆ Update your version of R to 3.2.5 or higher.
- ◆ Check your current version with *R.Version()*

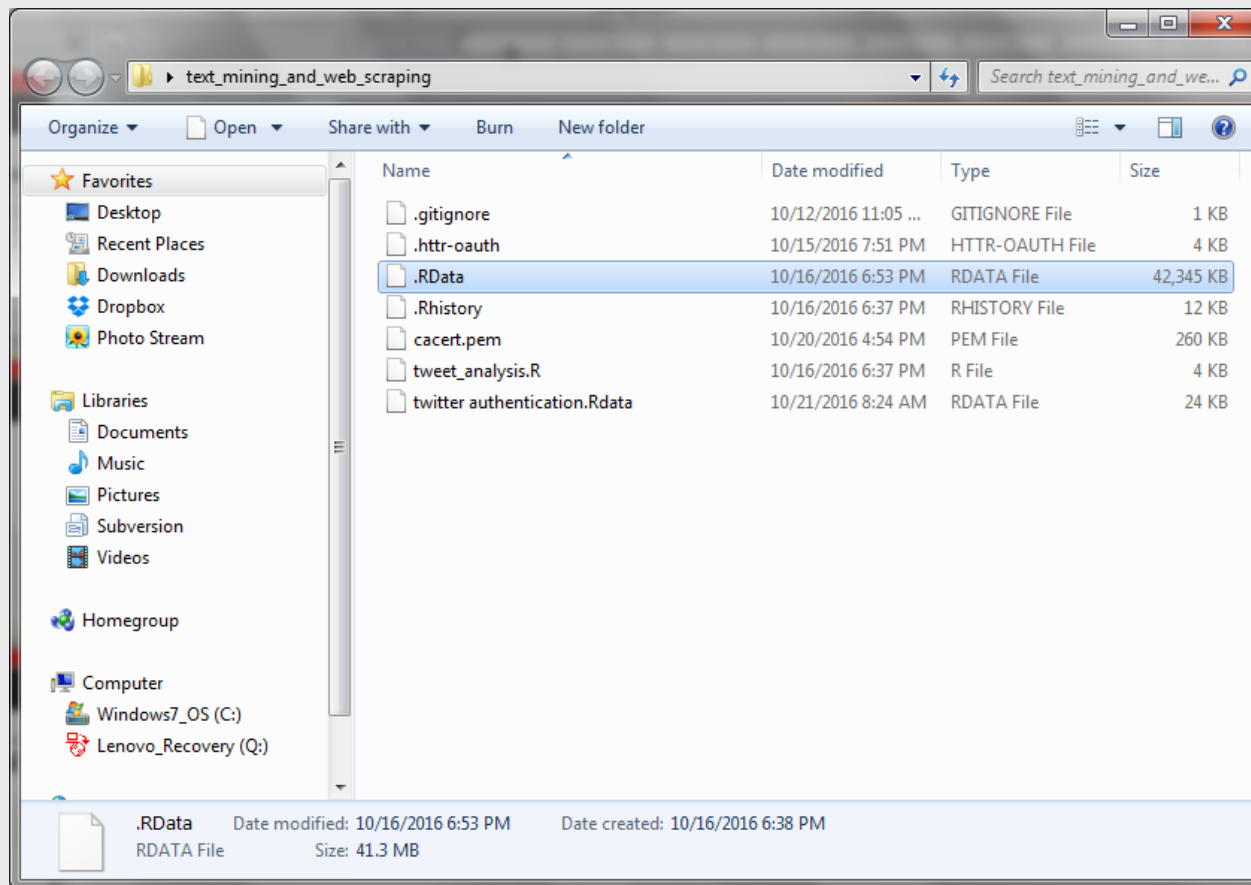


The screenshot shows a web browser window with the address bar displaying <https://cran.r-project.org/bin/windows/base/>. The page title is "Download R-3.3.1 for Windows (32/64 bit)". The main content area includes a link to "Download R 3.3.1 for Windows (70 megabytes, 32/64 bit)", followed by links for "Installation and other instructions" and "New features in this version". Below this, a paragraph states: "If you want to double-check that the package you have downloaded exactly matches the package distributed by R, you can compare the [md5sum](#) version of md5sum for windows: both [graphical](#) and [command line versions](#) are available." A section titled "Frequently asked questions" contains three bullet points: "Does R run under my version of Windows?", "How do I update packages in my previous version of R?", and "Should I run 32-bit or 64-bit R?". At the bottom, a note says: "Please see the [R FAQ](#) for general information about R and the [R Windows FAQ](#) for Windows-specific information." The footer of the page reads "Other builds".

Week 10 Topic:

Set up your working directory

- ◆ Check your working directory with *getwd()*
- ◆ Set your working directory with *setwd("C:/Users/~~~~")*



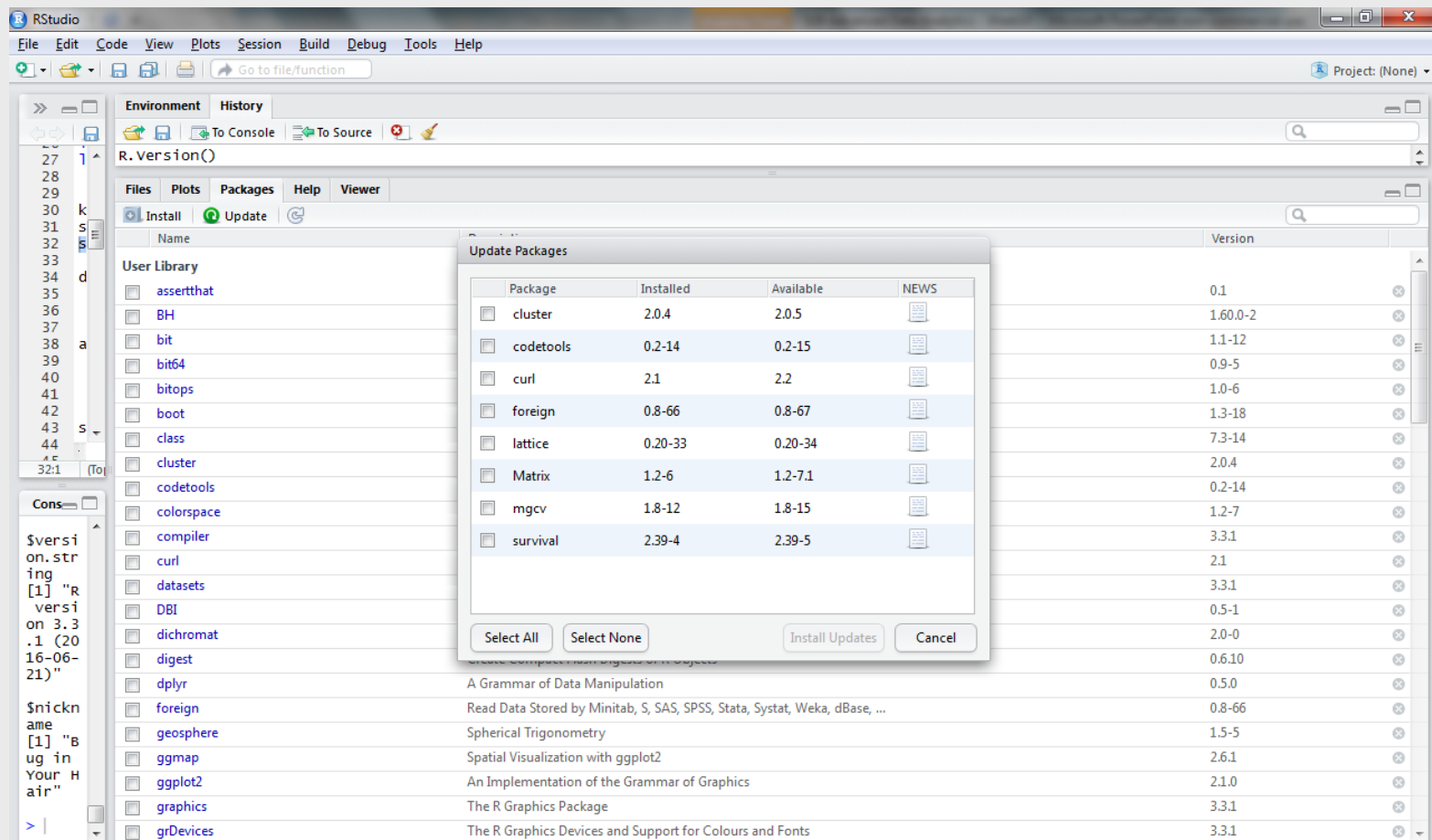
Week 10 Topic:

R Packages/Libraries Needed

- ◆ `install.packages("twitteR")`
- ◆ `install.packages("httpuv")`
- ◆ `install.packages("ROAuth")`
- ◆ `install.packages("stringr")`
- ◆ `install.packages("tm")`
- ◆ `install.packages("ggmap")`
- ◆ `install.packages("dplyr")`
- ◆ `install.packages("plyr")`
- ◆ `install.packages("wordcloud")`
- ◆ `install.packages("openssl")`
- ◆ `library(twitteR)`
- ◆ `library(httpuv)`
- ◆ `library(ROAuth)`
- ◆ `library(RCurl)`
- ◆ `library(stringr)`
- ◆ `library(tm)`
- ◆ `library(ggmap)`
- ◆ `library(plyr)`
- ◆ `library(dplyr)`
- ◆ `library(wordcloud)`
- ◆ `library(openssl)`

Week 10 Topic: Update R Packages/Libraries

- ◆ If you have packages installed previously, you can update packages:



The screenshot shows the RStudio interface with the 'Update Packages' dialog box open. The dialog lists the following packages and their versions:

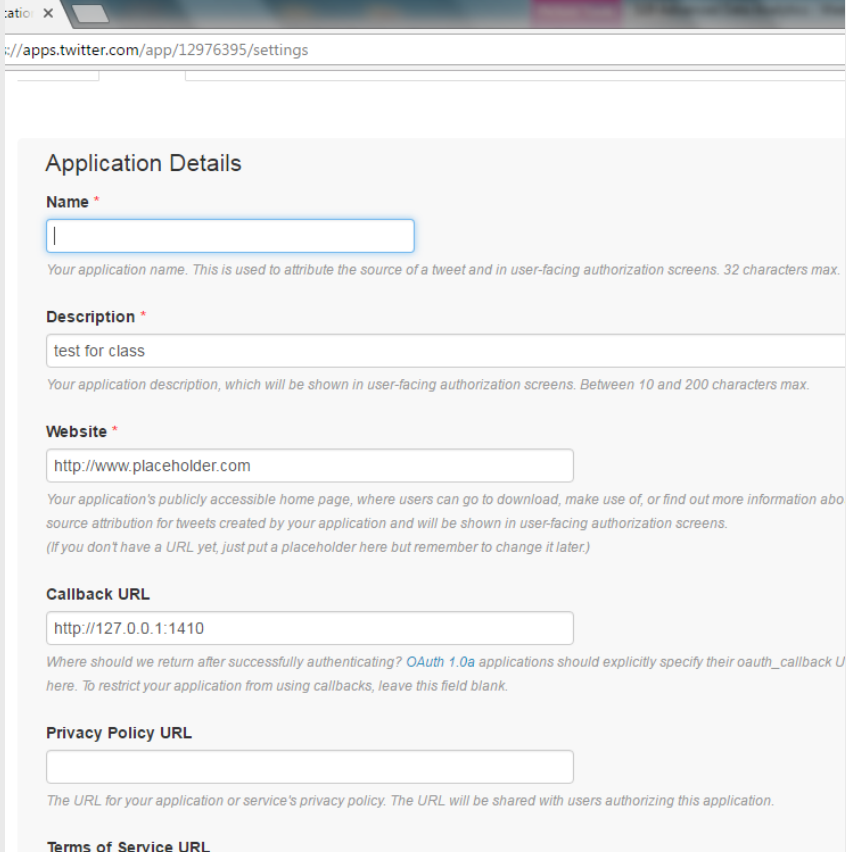
Package	Installed	Available	NEWS
cluster	2.0.4	2.0.5	0.1
codetools	0.2-14	0.2-15	1.60.0-2
curl	2.1	2.2	1.1-12
foreign	0.8-66	0.8-67	0.9-5
lattice	0.20-33	0.20-34	1.0-6
Matrix	1.2-6	1.2-7.1	1.3-18
mgcv	1.8-12	1.8-15	7.3-14
survival	2.39-4	2.39-5	2.0.4

The background shows the RStudio interface with the 'Packages' tab selected in the Environment pane. The 'User Library' section lists various installed packages, including assertthat, BH, bit, bit64, bitops, boot, class, cluster, codetools, colorspace, compiler, curl, datasets, DBI, dichromat, digest, dplyr, foreign, geosphere, ggmap, ggplot2, graphics, and grDevices.

Week 10 Topic:

Twitter Account/App set up

- ◆ In order to have access to Twitter data programmatically, one needs to create an app that interacts with the Twitter API.
- ◆ The first step is the registration of your app. In particular, you need to point your browser to <https://apps.twitter.com/> log-in to Twitter with a phone number (if you're not already logged in) and register a new application.
- ◆ Add the name and description of your app along with a website name. The website can be a test website.
- ◆ There's also a field for callback URL, but that's optional.



The screenshot shows the 'Application Details' form on the Twitter developer portal. The form includes fields for Name, Description, Website, Callback URL, Privacy Policy URL, and Terms of Service URL. The Name field is empty. The Description field contains 'test for class'. The Website field contains 'http://www.placeholder.com'. The Callback URL field contains 'http://127.0.0.1:1410'. The Privacy Policy URL and Terms of Service URL fields are empty. The form also includes instructions for each field, such as 'Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.'

Application Details

Name *

Description *

Website *

Callback URL

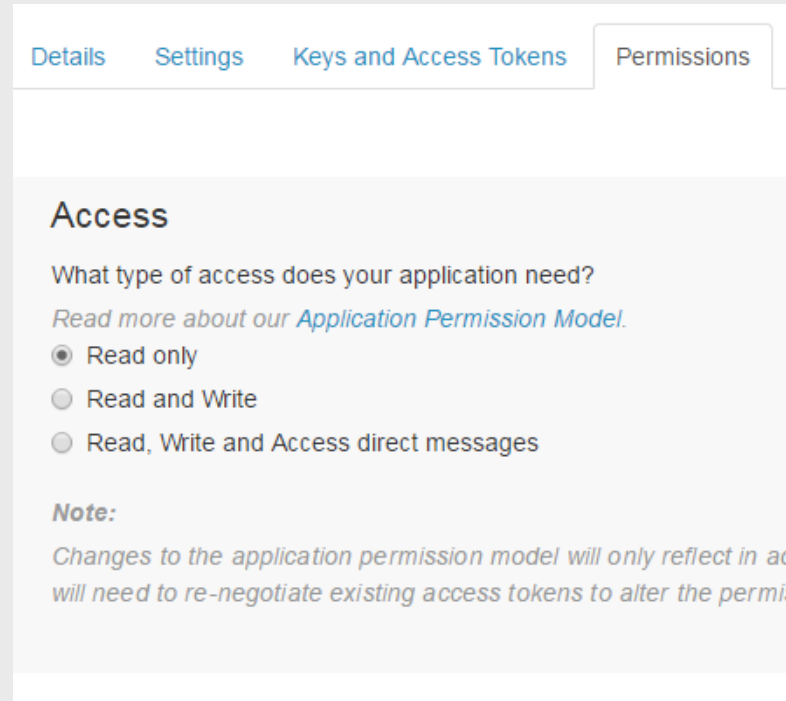
Privacy Policy URL

Terms of Service URL

Week 10 Topic:

Twitter Account/App set up (cont.)

- ◆ You will receive a *consumer key* and a *consumer secret*: these are application settings that should always be kept private.
- ◆ From the configuration page of your app, you can also require an access token and an access token secret. Similarly to the consumer keys, these strings must also be kept private: they provide the application access to Twitter on behalf of your account.
- ◆ The default permissions are read-only, which is all we need in our case, but if you decide to change your permission to provide writing features in your app, you must negotiate a new access token



The screenshot shows the 'Permissions' tab of the Twitter Developer Portal. At the top, there are four tabs: 'Details', 'Settings', 'Keys and Access Tokens', and 'Permissions'. The 'Permissions' tab is selected. Below the tabs, the section is titled 'Access'. It asks 'What type of access does your application need?' and provides a link to 'Read more about our Application Permission Model.' There are three radio button options: 'Read only' (which is selected), 'Read and Write', and 'Read, Write and Access direct messages'. Below these options is a 'Note:' section that states: 'Changes to the application permission model will only reflect in ac... will need to re-negotiate existing access tokens to alter the perm...'.

Week 10 Topic:

Authenticate App w Twitter

Grab your API keys and access tokens from Twitter:

```
>setup_twitter_oauth(api_key, api_secret, access_token, access_token_secret)
```

Or

```
>authenticate <- OAuthFactory$new(  
  consumerKey=key, consumerSecret=secret,  
  requestURL="https://api.twitter.com/oauth/request_token",  
  accessURL="https://api.twitter.com/oauth/access_token",  
  authURL="https://api.twitter.com/oauth/authorize")  
>setup_twitter_oauth(key, secret)
```

Save authentication:

```
>download.file(url="http://curl.haxx.se/ca/cacert.pem",  
  destfile="C:/Users/sshin/Desktop/text_mining_and_web_scraping/cacert.pem",  
  method="auto")  
>save(authenticate, file="twitter authentication.Rdata")
```

Week 10 Topic:

Grab Tweets, Create Library, Store

- ◆ # Grab latest tweets for Donal Trump and Hillary Clinton:

```
tweets_trump <- searchTwitter('@realDonaldTrump', n=1500)
```

```
tweets_clinton <- searchTwitter('@HillaryClinton', n=1500)
```

- ◆ Loop over tweets and extract text library(plyr):

```
feed_trump = laply(tweets_trump, function(t) t$getText())
```

```
feed_clinton= laply(tweets_clinton, function(t) t$getText())
```

- ◆ Write to csv as needed:

```
write.csv(feed_trump, "donaldtrump.csv",row.names = F)
```

```
write.csv(feed_clinton, "hilarlaryclinton.csv",row.names = F)
```

Week 10 Topic:

Cleanse Tweets

Now you've got a bunch of text data for Trump and Clinton, so how do we decide what's a "good" tweet and a "bad" tweet? This is where we turned to the [Hu and Liu Opinion Lexicon](#), a list of 6800 positive and negative words compiled by Bing Liu and Minqing Hu of the University of Illinois at Chicago.

- ◆ Unpack the Opinion Lexicon into your working directory.

Read in dictionary of positive and negative words

```
yay = scan('opinion-lexicon-English/positive-words.txt', what='character',  
comment.char=';')
```

```
boo = scan('opinion-lexicon-English/negative-words.txt', what='character',  
comment.char=';')
```

Add a few twitter-specific negative phrases

```
bad_text = c(boo, 'wtf', 'epicfail', 'douchebag')
```

```
good_text = c(yay, 'upgrade', ':)', '#iVoted', 'voted')
```

Week 10 Topic:

Scoring function

- ◆ Now, you've got your list of tweets and your list of opinionated words. The next thing to do is score the text of the tweets compared to how many of the "bad" and "good" words show up in each.
- ◆ For this we'll need a giant R function filled with lots of good gsub and match functions. Thanks to [Jeff Breen](#) for the function on which this was based:

```
score.sentiment = function(sentences, good_text, bad_text, .progress='none') {  
  require(plyr)  
  require(stringr)
```

```
# we got a vector of sentences. plyr will handle a list  
# or a vector as an "l" for us  
# we want a simple array of scores back, so we use  
# "l" + "a" + "ply" = "lapply":  
, text=sentences) return(scores.df) }
```

Week 10 Topic:

Scoring function (cont.)

```
scores = laply(sentences, function(sentence, good_text, bad_text) {  
  
  # clean up sentences with R's regex-driven global substitute, gsub():  
  sentence = gsub('[:punct:]]', '', sentence)  
  sentence = gsub('[:cntrl:]]', '', sentence)  
  sentence = gsub('\\ \\d+', '', sentence)  
  
  #to remove emojis  
  sentence <- iconv(sentence, 'UTF-8', 'ASCII')  
  
  # and convert to lower case:  
  sentence = tolower(sentence)
```

Week 10 Topic:

Scoring function (cont.)

```
# split into words. str_split is in the stringr package  
word.list = str_split(sentence, '\\s+')
```

```
# sometimes a list() is one level of hierarchy too much  
words = unlist(word.list)
```

```
# compare our words to the dictionaries of positive & negative terms  
pos.matches = match(words, good_text)  
neg.matches = match(words, bad_text)
```

```
# match() returns the position of the matched term or NA  
# we just want a TRUE/FALSE:  
pos.matches = !is.na(pos.matches)  
neg.matches = !is.na(neg.matches)  
)}
```

Week 10 Topic:

Scoring function (cont.)

```
# and conveniently enough, TRUE/FALSE will be treated as 1/0 by sum():  
score = sum(pos.matches) - sum(neg.matches)  
return(score) }, good_text, bad_text, .progress=.progress )
```

```
scores.df = data.frame(score=scores, text=sentences)  
return(scores.df) }
```

Week 10 Topic:

Calling and Plotting

Call the function and return a data frame

```
feelthatrump <- score.sentiment(feed_trump, good_text, bad_text, .progress='text')
```

```
feelthaclinton <- score.sentiment(feed_clinton, good_text, bad_text, .progress='text')
```

Nice little quick plot

```
qplot(factor(score), data=feelthatrump, geom="bar", xlab = "Sentiment Score")
```

```
qplot(factor(score), data=feelthaclinton, geom="bar", xlab = "Sentiment Score")
```


Week 10 Topic: Reference

- ◆ <https://sites.google.com/site/miningtwitter/questions/sentiment/analysis>:

basic analysis - Mining T... X

← → ↻ <https://sites.google.com/site/miningtwitter/questions/sentiment/analysis> ★

Mining twitter with

Search this site

- Home
- Intro
- ▼ Mining Basics
 - 1 getting data
 - 2 processing
 - 3 text mining
- ▼ Mining Questions
 - 1 talking about
 - 2 frequencies
 - 3 user tweets
 - 4 sentiments
- Mining Viz
- References
- GitHub

Basic Sentiment Analysis in R [sentiments](#)

Breen's Approach

One option to perform sentiment analysis in R is by following what I call the *Breen's approach* named after Jeffrey Breen's seminal elucidating slides on twitter sentiment analysis with R <http://jeffreymbreen.wordpress.com/2011/07/04/twitter-text-mining-r-slides/>

The general idea is to calculate a *sentiment score* for each tweet so we can know how positive or negative is the posted message. There are different ways to calculate such scores, and you can even create your own formula. We'll use a very simple yet useful approach to define our score formula

$$\text{Score} = \text{Number of positive words} - \text{Number of negative words}$$

If Score > 0, this means that the sentence has an overall 'positive opinion'
If Score < 0, this means that the sentence has an overall 'negative opinion'
If Score = 0, then the sentence is considered to be a 'neutral opinion'

In order to count the number of positive and negative words, we need a very important ingredient: an opinion lexicon in english, which fortunately it is provided by [Hu and Liu](#) and it can be accessed from: <http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html>

Minqing Hu and Bing Liu. "Mining and Summarizing Customer Reviews." *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004)*, Aug 22-25, 2004, Seattle, Washington, USA,

Bing Liu, Minqing Hu and Junsheng Cheng. "Opinion Observer: Analyzing and Comparing Opinions on the Web." *Proceedings of the 14th*

STAC-A2 Report I...pdf ^ STAC-A2 Report I...docx ^ STAC-A2 Configu...docx ^ Show all X

Week 10 Topic:

Week 10 Assignment

Develop your own sentiment analysis scoring function:

- 1) Customize/Augment/Update the “good text”, “bad text” repository with own research/list of words. Survey the tweets (in XLS or other) and determine words to be considered for either. Submit a table of “good” and “bad” words.
- 2) Determine additional levels of scoring e.g., more than just good and bad, weighting of words by importance, weighting of words by frequency etc. Submit scoring logic in a table.
- 3) Develop final scoring method and question. Submit the final scoring equation and function.
- 4) Share in discussion topic

Week 11 Topic:

Agenda

Sentiment Analysis of Tweets:

- ◆ Week 10-11:
 - Create a Dev account on Twitter
 - Develop own sentiment scoring function
 - Score and plot results - On ~2k tweets
- ◆ Week 12: Visualize analytics in Tableau
 - Collect geo data for tweets
 - Store tweets in corpus
 - Plot in Tableau
- ◆ Week 13: Store tweets in Hadoop
 - Score and plot results - On ~10k tweets
- ◆ Week 14: Store tweets in MongoDB
 - Score and plot results - On ~10k tweets

Week 11 Topic:

References

Use following examples as reference for the coming weeks:

NLP etc.:

- ◆ Unsupervised: <http://stackoverflow.com/questions/3920759/unsupervised-sentiment-analysis>
- ◆ NLP Tools: <http://stackoverflow.com/questions/12299724/list-of-natural-language-processing-tools-in-regards-to-sentiment-analysis-whi?rq=1>

Weeks 10-12:

- ◆ Donald Trump: <https://www.r-bloggers.com/sentiment-analysis-on-donald-trump-using-r-and-tableau/>
- ◆ Super Tuesday: <https://www.r-bloggers.com/how-to-use-r-to-scrape-tweets-super-tuesday-2016/>
- ◆ Jazz: <https://jazzanalytics.wordpress.com/2016/09/12/sentiment-analysis-on-narendra-modi-using-r/>
- ◆ Basic Sentiment Example: <https://sites.google.com/site/miningtwitter/questions/sentiment/analysis>

Week 11 Topic: References (cont.)

Weeks 10-12 (cont.):

- ◆ Vectorizing: <http://stackoverflow.com/questions/25184076/using-scores-in-sentiment-analysis-with-r?rq=1>
- ◆ Various packages: <http://stackoverflow.com/questions/10233087/sentiment-analysis-using-r?rq=1>
- ◆ Other word lists: <http://stackoverflow.com/questions/1196133/seed-data-for-sentiment-analysis?rq=1>
- ◆ Other approaches: <http://stackoverflow.com/questions/4199441/best-algorithmic-approach-to-sentiment-analysis?rq=1>
- ◆ Scoring by Sentence: <https://blog.exploratory.io/twitter-sentiment-analysis-scoring-by-sentence-b4d455de3560#.c7ofvrcjn>
- ◆ 3 lists example: <http://analyzecore.com/2014/04/28/twitter-sentiment-analysis/>
- ◆ <https://sites.google.com/site/miningtwitter/basics/processing>
- ◆ <https://cran.r-project.org/web/packages/twitteR/twitteR.pdf>

Week 11 Topic:

Language References

Language:

- ◆ Phrase Structure Rules: https://en.wikipedia.org/wiki/Phrase_structure_rules
- ◆ Grammar Frameworks:
https://en.wikipedia.org/wiki/Category:Grammar_frameworks

Presentations on sentiment analysis and datamining:

- ◆ <http://condor.depaul.edu/ntomuro/courses/594/>

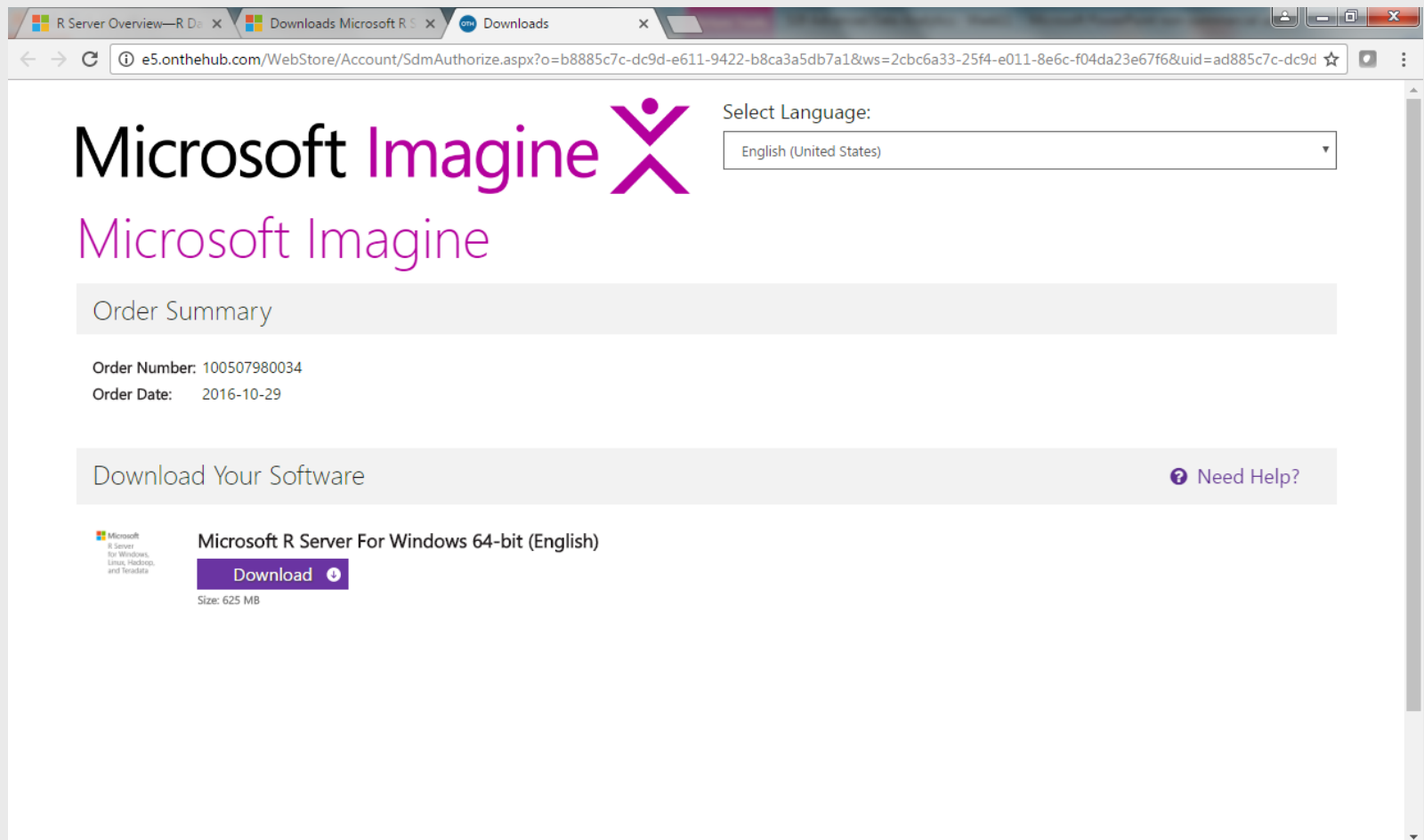
Book:

- ◆ Text Mining and Analysis: Practical Methods, Examples, and Case Studies Using SAS:
<http://proquestcombo.safaribooksonline.com/book/databases/sas/9781612907871>

Just a fun app:

- ◆ <http://emotify.accessible.ai/>

Week 11 Topic: Microsoft Dreamspark – R Server



The screenshot shows a web browser window with the URL `e5.onthehub.com/WebStore/Account/SdmAuthorize.aspx?o=b8885c7c-dc9d-e611-9422-b8ca3a5db7a1&ws=2cbc6a33-25f4-e011-8e6c-f04da23e67f6&uid=ad885c7c-dc9d`. The page features the Microsoft Imagine X logo and the text "Microsoft Imagine". A "Select Language:" dropdown menu is set to "English (United States)". Below this, there is an "Order Summary" section with the following details:

- Order Number: 100507980034
- Order Date: 2016-10-29

The "Download Your Software" section includes a "Need Help?" link. The software being downloaded is "Microsoft R Server For Windows 64-bit (English)", with a "Download" button and a size of 625 MB.

Week 11 Topic:

Free Trial - Google Natural Language API

CLOUD NATURAL LANGUAGE API ^{BETA}

https://cloud.google.com/natural-language/?utm_source=google&utm_medium=cpc&utm_campaign=2015-q2-cloud-na-gcp-bkws-freetrial-en&gclid=CPap-beU_s8CFQ6TaQod8dcAPg

The screenshot shows the Google Cloud Platform documentation page for the Natural Language API. The browser tabs at the top include 'API Manager - My First P...', 'Google APIs Explorer', and 'Google Cloud Natural La...'. The address bar shows the URL: <https://cloud.google.com/natural-language/docs/>. The page header features the Google Cloud Platform logo, a search bar, and a 'Console' button. A navigation menu includes links for 'Why Google', 'Products', 'Solutions', 'Launcher', 'Pricing', 'Customers', 'Documentation' (which is highlighted), 'Support', 'Partners', and a 'Contact Sales' button. The main content area is titled 'Google Cloud Natural Language API Documentation' and includes a star rating. The left sidebar contains a table of contents with sections: 'Natural Language API' (with links for Product Overview, Documentation, and Quickstart), 'How-to Guides' (with link for Authenticating to the API), 'APIs & Reference' (with links for REST API and v1beta1), 'Concepts' (with link for Natural Language Basics), 'Samples & Tutorials' (with links for All Samples & Tutorials, Sentiment Analysis Tutorial, and Sample Applications), and 'Resources' (with links for All Resources, Support, and Limits). The main content area provides an overview of the API, stating it provides natural language understanding technologies to developers, including sentiment analysis, entity recognition, and syntax analysis. It also mentions that the API currently supports English for sentiment analysis and English, Spanish, and Japanese for entity analysis and syntax analysis. At the bottom, there are six tiles for 'Quickstart', 'How-to Guides', 'APIs & Refere...', 'Concepts', 'Samples & Tu...', and 'Resources', each with a brief description and a link.

Google Cloud Platform

Why Google Products Solutions Launcher Pricing Customers Documentation Support Partners Contact Sales

Natural Language API

- Product Overview
- Documentation
- Quickstart

How-to Guides

- Authenticating to the API

APIs & Reference

- REST API
- v1beta1

Concepts

- Natural Language Basics

Samples & Tutorials

- All Samples & Tutorials
- Sentiment Analysis Tutorial
- Sample Applications

Resources

- All Resources
- Support
- Limits

Google Cloud Natural Language API

Google Cloud Natural Language API Documentation

The Google Cloud Natural Language API provides natural language understanding technologies to developers, including sentiment analysis, entity recognition, and syntax analysis. This API is part of the larger Cloud Machine Learning API.

The Cloud Natural Language API currently supports English for [sentiment analysis](#) and English, Spanish, and Japanese for [entity analysis](#) and [syntax analysis](#).

Quickstart
Learn in 5 minutes

How-to Guides
Perform specific tasks

APIs & Refere...
JSON Reference

Concepts
Develop a deep understanding of Cloud Natural Language API

Samples & Tu...
Walkthroughs and sample applications

Resources
Pricing, quotas, release notes, and other resources

Week 11 Topic: Developing Good R Code

<https://cran.r-project.org/web/packages/rockchalk/vignettes/Rchaeology.pdf>:

Rchaeology: Idioms of R Programming

Paul E. Johnson <pauljohn @ ku.edu>

February 24, 2016

This document was initiated on May 31, 2012. The newest copy will always be available at <http://pj.freefaculty.org/R> and as a vignette in the R package “rockchalk”.

Rchaeology: The study of R programming by investigation of R source code. It is the effort to discern the programming strategies, idioms, and style of R programmers in order to better communicate with them.

Rchaeologist: One who practices Rchaeology.

These are Rcheological observations about the style and mannerisms of R programmers in their native habitats. Almost all of the insights here are gathered from the r-help and r-devel emails lists, the stackoverflow website pages for R, and the R source code itself. These are lessons from the “school of hard knocks.”

How is this different from Rtips(<http://pj.freefaculty.org/R/Rtips.{pdf,html}>)?

1. This is oriented toward programming R, rather than using R.
2. It is more synthetic, aimed more at finding “what’s right” rather than “what works.”

Week 11 Topic:

do.call, lapply, and map

<http://stackoverflow.com/questions/10801750/whats-the-difference-between-lapply-and-do-call-in-r>:

“In most simple words:

- ◆ ***lapply()*** applies a given function for each element in a list, so there will be several function calls.
- ◆ ***do.call()*** applies a given function to the list as a whole, so there is only one function call.

The best way to learn is to play around with the function examples in the R documentation.”

- ◆ Do.call RDocumentation:
<https://www.rdocumentation.org/packages/base/versions/3.3.1/topics/do.call>
- ◆ lapply RDocumentation:
<https://www.rdocumentation.org/packages/base/versions/3.3.1/topics/lapply>
- ◆ Map Rdocumentation:
<https://www.rdocumentation.org/packages/lambda.tools/versions/1.0.9/topics/map>

Week 11 Topic:

Understanding do.call

<http://www.stat.berkeley.edu/~s133/Docall.html>:

- ◆ R has an interesting function called `do.call`. This function allows you to call any R function, but instead of writing out the arguments one by one, you can use a list to hold the arguments of the function. While it may not seem useful on the surface, a simple example will help to show how powerful `do.call` is.
- ◆ Suppose we have three comma-separated text files that have information on the same three variables.
- ◆ It's easy to read each one in with `read.csv`, and then to call the `rbind` (combine by rows) function to make one big data frame. (Remember that `rbind` will only work if all the data frames being combined have the same variable names.)

```
> one = read.csv('1.csv')
> nrow(one)
[1] 21
> two = read.csv('2.csv')
> nrow(two)
[1] 25
> three = read.csv('3.csv')
> nrow(three)
[1] 27
> big = rbind(one,two,three)
> nrow(big)
[1] 73
```

Week 11 Topic:

Understanding do.call (cont)

- ◆ But now suppose we have 20 csv files that we want to read and combine. We could do what we did with the three files, but, not only would it get tiring, there's a chance of making an error when we have to type so many commands.
- ◆ In the past, when we've had problems like this, *sapply* was able to help. Let's try it here, by writing a function that will take a number, create a filename by pasting .csv at the end, and reading in the data:

```
> allframes = sapply(1:20,function(x)read.csv(paste(x,'csv',sep='.')))
> head(allframes)
[,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]
a factor,21 factor,25 factor,27 factor,25 factor,27 factor,21 factor,24
b factor,21 factor,25 factor,27 factor,25 factor,27 factor,21 factor,24
x Numeric,21 Numeric,25 Numeric,27 Numeric,25 Numeric,27 Numeric,21 Numeric,24
y Numeric,21 Numeric,25 Numeric,27 Numeric,25 Numeric,27 Numeric,21 Numeric,24
[,8]  [,9]  [,10] [,11] [,12] [,13] [,14]
a factor,28 factor,23 factor,23 factor,22 factor,26 factor,24 factor,23
b factor,28 factor,23 factor,23 factor,22 factor,26 factor,24 factor,23
x Numeric,28 Numeric,23 Numeric,23 Numeric,22 Numeric,26 Numeric,24 Numeric,23
```

Week 11 Topic:

Understanding do.call (cont)

- ◆ It will always return a list, the same length as its first argument, with each element of the list resulting in the function we passed to lapply operating on one element of the first argument.
- ◆ In our case, calling lapply instead of sapply will give us a list of length 20, where each element is the result of calling read.csv on one of the 20 files. This is where do.call comes in. Instead of having to pass 20 data frames to rbind, we can use do.call to pass all 20 of them to rbind, since they are in a list, and that's exactly what do.call is looking for.

```
> allframes = lapply(1:20,function(x)read.csv(paste(x,'csv',sep='.')))  
> sapply(allframes,nrow)  
[1] 21 25 27 25 27 21 24 28 23 23 22 26 24 23 25 29 28 30 27 29  
> answer = do.call(rbind,allframes)  
> nrow(answer)  
[1] 507
```

We can combine all the data frames without storing them separately or passing them individually to rbind.

Week 11 Topic:

Get List of Tweets and Geo Locs

```
>N=2000 # tweets to request from each query
>S=200 # radius in miles
>lats=c(38.9,40.7,37.8,39,37.4,28,30,42.4,48,36,32.3,33.5,34.7,33.8,37.2,41.2,
46.8, 46.6,37.2,43,42.7,40.8,36.2,38.6,35.8,40.3,43.6,40.8,44.9,44.9)
>lons=c(-77,-74,-122,-105.5,-122,-82.5,-98,-71,-122,-115,-86.3,-112,-92.3,-
84.4,-93.3, -104.8,-100.8,-112, -93.3,-89,-84.5,-111.8,-86.8,-92.2,-78.6,-76.8,-
116.2,-98.7,-123,-93)
```

#cities=DC,New York,San Fransisco, Colorado, Mountainview, Tampa, Austin, Boston, Seatle,Vegas,Montgomery,Phoenix,Little Rock,Atlanta,Springfield, Cheyenne,Bisruk,Helena,Springfield,Madison,Lansing,Salt Lake City,Nashville, Jefferson City,Raleigh,Harrisburg,Boise,Lincoln,Salem,St. Paul

```
>donald=do.call(rbind,lapply(1:length(lats), function(i)
searchTwitter('Donald+Trump', lang="en",n=N, resultType="recent",
geocode=paste(lats[i],lons[i],paste0(S,"mi"), sep=","))))
```

Week 11 Topic:

Get tweets into DF

```
>donaldlat=sapply(donald, function(x) as.numeric(x$getLatitude()))
>donaldlat=sapply(donaldlat, function(z) ifelse(length(z)==0,NA,z))
>donaldlon=sapply(donald, function(x) as.numeric(x$getLongitude()))
>donaldlon=sapply(donaldlon, function(z) ifelse(length(z)==0,NA,z))
>donalddate=lapply(donald, function(x) x$getCreated())
>donalddate=sapply(donalddate,function(x) strftime(x, format="%Y-%m-%d
%H:%M:%S",tz = "UTC")) donaldtext=sapply(donald, function(x) x$getText())
>donaldtext=unlist(donaldtext)
>isretweet=sapply(donald, function(x) x$getIsRetweet())
>retweeted=sapply(donald, function(x) x$getRetweeted()) >retweetcount=sapply(donald,
function(x) x$getRetweetCount()) >favoritecount=sapply(donald, function(x)
x$getFavoriteCount()) >favorited=sapply(donald, function(x) x$getFavorited())
```

```
>data=as.data.frame(cbind(tweet=donaldtext,date=donalddate,lat=donaldlat,lon=donaldlon,
isretweet=isretweet,retweeted=retweeted,
retweetcount=retweetcount,favoritecount=favoritecount,favorited=favorited))
```

Week 11 Topic:

Examine the List

```
>length(donaldlon)
>write.csv(donaldlon, "donaldlon.csv",row.names = F)
>length(donaldlat)
>write.csv(donaldlat, "donaldlat.csv",row.names = F)
>length(donalddate)
>write.csv(donalddate, "donalddate.csv",row.names = F)
>length(donaldtext)
write.csv(donaldtext, "donaldtext.csv",row.names = F)

>ls()
```


Week 11 Topic:

Using twListToDF

References:

- ◆ <https://www.rdocumentation.org/packages/twitteR/versions/1.1.9/topics/twListToDF>
- ◆ <http://rfunction.com/archives/2002>

```
> library(twitteR)
Loading required package: RCurl
Loading required package: bitops
Loading required package: RJSONIO
> result <- userTimeline("BarackObama", n=3200)
> length(result)
[1] 2975
> tweet.df <- twListToDF(result)
> Created <- tweet.df$created
> counts <- table(as.Date(Created))[-1]
> dates <- as.Date(names(counts))
> # png("twitteR-BarackPosts.png", 650, 500)
> plot(dates, counts, type="h") > # dev.off()
```

Week 11 Topic:

Corpus Word Cloud

```
# Create corpus
```

```
>corpus=Corpus(VectorSource(data$tweet))
```

```
# Convert to lower-case
```

```
>corpus=tm_map(corpus,tolower)
```

```
# Remove stopwords
```

```
>corpus=tm_map(corpus,function(x) removeWords(x,stopwords()))
```

```
# convert corpus to a Plain Text Document
```

```
>corpus=tm_map(corpus,PlainTextDocument)
```

```
>col=brewer.pal(6,"Dark2")
```

```
>wordcloud(corpus, min.freq=25, scale=c(5,2),rot.per = 0.25, random.color=T,  
max.word=45, random.order=F,colors=col)
```

Week 11 Topic: SAS Enterprise Miner

<https://support.sas.com/ondemand/emConfig.html>

The screenshot shows a web browser window with the URL <https://support.sas.com/ondemand/emConfig.html>. The page is titled "Step Three: Install the recommended Java Runtime Environment and Java Web Start". It provides instructions for installing the recommended JRE, including a list of steps and a note about alternative JREs. The steps are:

1. Access the [Java SE Downloads](#) Web Site and then download a JRE from the 1.8 series that is either before Update 72 or is Update 73. For example, to download JRE 1.8.0_45, do the following:
 1. Open the [Oracle Java Archive](#) site.
 2. Click [Java SE 8](#).
 3. Click [Java SE Runtime Environment 8u45](#).
 4. Check the [Accept License Agreement](#) radio button to accept the Oracle Binary Code License Agreement for Java SE.
 5. Select [jre-8u45-windows-i586.exe](#) from the [Download](#) column of the table.
 6. Launch the download and select [Run](#) to begin the installation process.
2. (Optional) If you had to complete [Step Two](#) above, repeat this step to ensure that only the new JRE that you installed is selected.

Notes:

- Other JREs from the 1.7 and 1.8 series might also work.
- For more information, see [SAS® 9.4 Support for Java Runtime Environments](#).

[RETURN TO TOP](#)

Starting SAS® OnDemand for Academics: Enterprise Miner™

After you have installed the recommended JRE (or if you want to test with a JRE that is already installed), you can [start SAS® OnDemand for Academics: Enterprise Miner™](#).

In the future, you can simply [start SAS® OnDemand for Academics: Enterprise Miner™](#) any time that you want to use the application.

Note: If you experience problems starting the application, consider [clearing your Java cache](#).

The bottom of the screenshot shows a Windows taskbar with a security warning: "This type of file can harm your computer. Do you want to keep main.jsp anyway?" and buttons for "Keep" and "Discard". The taskbar also shows several open files: "Japan_US_GameR....pptx", "TM-10-Sentiment....ppt", and "ACRE_Install.zip".

Week 11 Topic:

Adjusting sentiment scoring for phrases

<http://stackoverflow.com/questions/28545270/sentiment-score-for-sentence-in-r>

Stack Overflow question: <http://stackoverflow.com/questions/28545270/sentiment-score-for-sentence-in-r>

Answer by user 1:

I didn't look at the library you mentioned. This may now be what you want. I created a data frame with the positive and negative words. I assigned them a +/- 1 value. I then assigned them a length value to sort on. This way the longest word/phrase is used first.

```
sent <- data.frame(words = c("just right size", "love this quality",
                             "good quality", "very good quality", "i hate this notebook",
                             "great improvement", "notebook is not good"), user = c(1,2,3,4,
                             stringsAsFactors=F)

posWords <- c("great","improvement","love","great improvement","very good","good","right","v
negWords <- c("hate","bad","not good","horrible")

wordsDF<- data.frame(words = posWords, value = 1,stringsAsFactors=F)
wordsDF<- rbind(wordsDF,data.frame(words = negWords, value = -1))
wordsDF$lengths<-unlist(lapply(wordsDF$words, nchar))
wordsDF<-wordsDF[ order(-wordsDF[,3]),]

scoreSentence <- function(sentence){
  score<-0
  for(x in 1:nrow(wordsDF)){
    count<-length(grep(wordsDF[x,1],sentence))
    if(count){
      score<-score + (count * wordsDF[x,2])
      sentence<-sub(wordsDF[x,1],'',sentence)
    }
  }
  score
}

SentimentScore<- unlist(lapply(sent$words, scoreSentence))
cbind(sent, SentimentScore)
```

Output

	words	user	SentimentScore
1	just right size	1	1

Hot Network Questions

- proper oil for 1940's era aluminum ford engine
- Is there a name for the (anti-) pattern of passing parameters that will only be used several levels deep in the call chain?
- Most environmentally friendly mode of transport
- How much more than my mortgage should I charge for rent?
- If I receive written permission to use, without citing, a paper, is it plagiarism?
- Why didn't Japan attack the West Coast of the United States during World War II?
- Why is this C++ code faster than my hand-written assembly?
- Interlace strings
- Trick or Treat polyglot
- Client requesting admin work
- C++ Initialising fields directly vs initialisation list in default constructor
- Four color theorem disproof?
- Power cycling old hardware
- Why are spare wheels smaller than normal wheels?
- Does Dispel Magic end Spiritual Weapon?
- Were there any U.S. Presidents that were very unpopular when elected
- Is it accurate that the median is a score where 50% of the observations are above it and 50% are below it?

Week 11 Topic:

Adjusting sentiment scoring code

```
sent <- data.frame(words = c("just right size", "love this quality", "good quality", "very good quality", "i hate this notebook", "great improvement", "notebook is not good"), user = c(1,2,3,4,5,6,7), stringsAsFactors=F)
posWords <- c("great","improvement","love","great improvement","very good","good","right","very")
negWords <- c("hate","bad","not good","horrible")
```

```
wordsDF<- data.frame(words = posWords, value = 1,stringsAsFactors=F)
wordsDF<- rbind(wordsDF,data.frame(words = negWords, value = -1))
wordsDF$lengths<-unlist(lapply(wordsDF$words, nchar))
wordsDF<-wordsDF[ order(-wordsDF[,3]),]
```

```
scoreSentence <- function(sentence){
  score<-0
  for(x in 1:nrow(wordsDF)){
    count<-length(grep(wordsDF[x,1],sentence))
    if(count){
      score<-score + (count * wordsDF[x,2])
      sentence<-sub(wordsDF[x,1],",",sentence) } }
  score }
```

```
SentimentScore<- unlist(lapply(sent$words, scoreSentence))
cbind(sent, SentimentScore)
```

Week 11 Topic:

R Tips

<https://www.zoology.ubc.ca/~schluter/R/data/>

The screenshot shows a web browser window displaying the 'R tips pages' website. The browser's address bar shows the URL <https://www.zoology.ubc.ca/~schluter/R/data/>. The website has a dark blue header with the R logo and the text 'R tips pages'. Below the header is a navigation bar with links: Start, Vector, Data, Display, Plan, Loop, Fit model, Prob, Resample, Meta-analysis, Multivar, Phylo, Biol 501, and Workshops. The 'Data' link is highlighted. The main content area is titled 'Data' and contains the following text: 'This page introduces the basics of working with data sets having multiple variables, often of several types. The focus here is on data frames, which are the most convenient data objects in R. Matrices and lists are also useful data objects, and these are introduced briefly at the end.' Below this text is a section titled 'Manage data' with the subheading 'Start with a spreadsheet program'. The text in this section states: 'It is best to enter your data to an ordinary text file, such as a .csv (comma-separated text) file, created with the help of a spreadsheet program such as Excel or the free program Calc. A text file is never obsolete, whereas a proprietary format may not be readable 10 years from now. Avoid special characters in your data files.' Below this is a section titled 'Long vs wide layouts' with the text: 'Keep data that you want analyzed together in a single worksheet. A "long" layout is recommended, rather than a wide layout. Here is an example of a wide layout of data on the numbers of individuals of 3 species recorded in plots and sites.' At the bottom of the page, there is a table with the following columns: Plot, Site, species1, species2, and species3. The table is partially visible, showing the first few rows of data.

Data

This page introduces the basics of working with data sets having multiple variables, often of several types. The focus here is on data frames, which are the most convenient data objects in R. Matrices and lists are also useful data objects, and these are introduced briefly at the end.

Manage data

Start with a spreadsheet program

It is best to enter your data to an ordinary text file, such as a .csv (comma-separated text) file, created with the help of a spreadsheet program such as Excel or the free program Calc. A text file is never obsolete, whereas a proprietary format may not be readable 10 years from now. Avoid special characters in your data files.

Long vs wide layouts

Keep data that you want analyzed together in a single worksheet. A "long" layout is recommended, rather than a wide layout. Here is an example of a wide layout of data on the numbers of individuals of 3 species recorded in plots and sites.

Plot	Site	species1	species2	species3

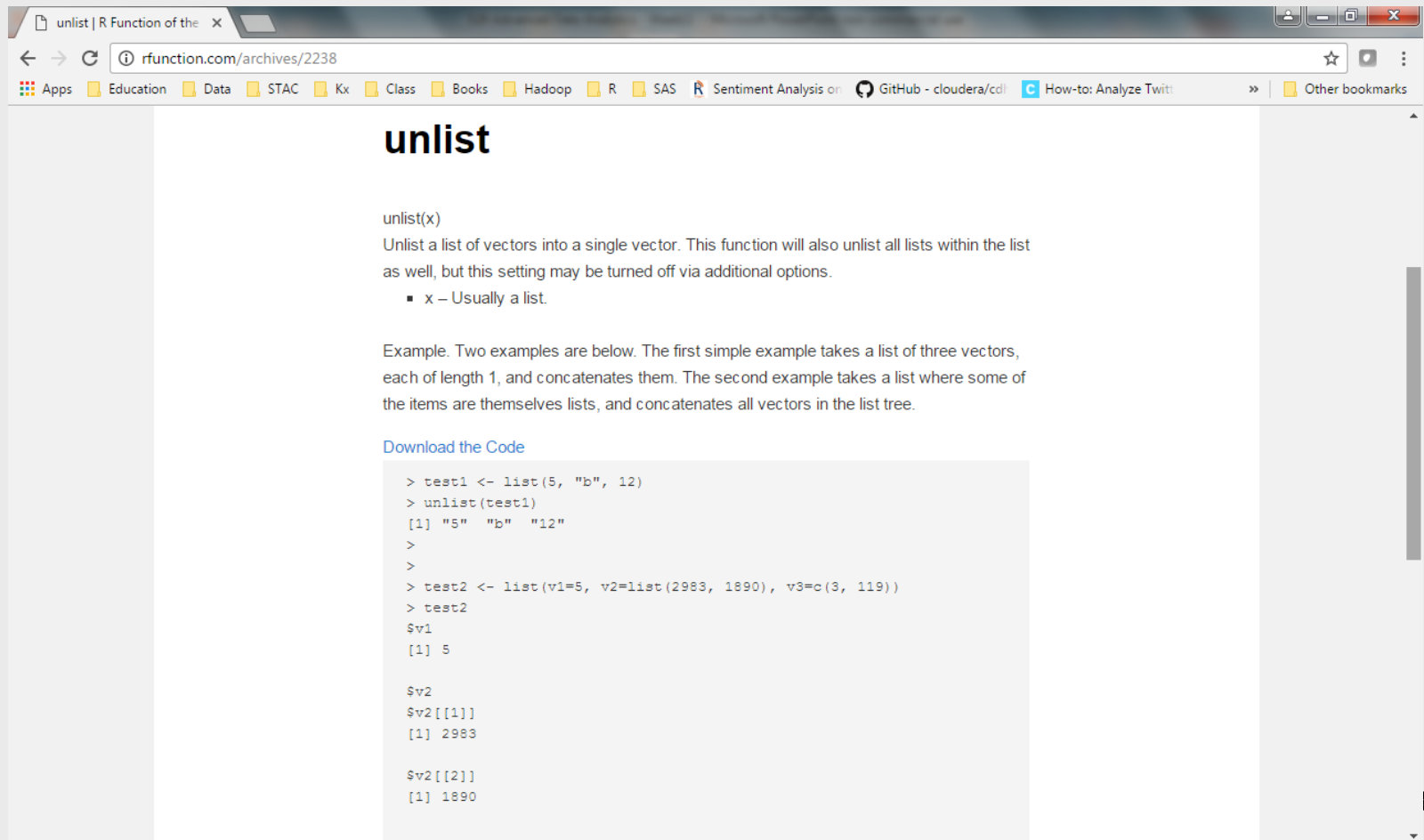
Navigate data frame page

- Manage data
- Read data from file
- Write data frame to file
- Work with data frames
 - View contents
 - Data frame functions
 - Access variables
 - Transform variable
 - Apply function to variable
 - Delete variable
 - Combine vectors
- Sort and order a data frame
- Extract subset of data frame
- Match 2 data frames
- Attach a data frame
- Other data structures: matrix

Week 11 Topic:

unlist()

<http://rfunction.com/archives/2238>



The screenshot shows a web browser window with the URL <http://rfunction.com/archives/2238>. The page title is "unlist | R Function of the". The browser's address bar shows the URL. The page content includes a heading "unlist", a description of the function, and an example of its usage.

unlist

unlist(x)

Unlist a list of vectors into a single vector. This function will also unlist all lists within the list as well, but this setting may be turned off via additional options.

- x – Usually a list.

Example. Two examples are below. The first simple example takes a list of three vectors, each of length 1, and concatenates them. The second example takes a list where some of the items are themselves lists, and concatenates all vectors in the list tree.

[Download the Code](#)

```
> test1 <- list(5, "b", 12)
> unlist(test1)
[1] "5" "b" "12"
>
>
> test2 <- list(v1=5, v2=list(2983, 1890), v3=c(3, 119))
> test2
$v1
[1] 5

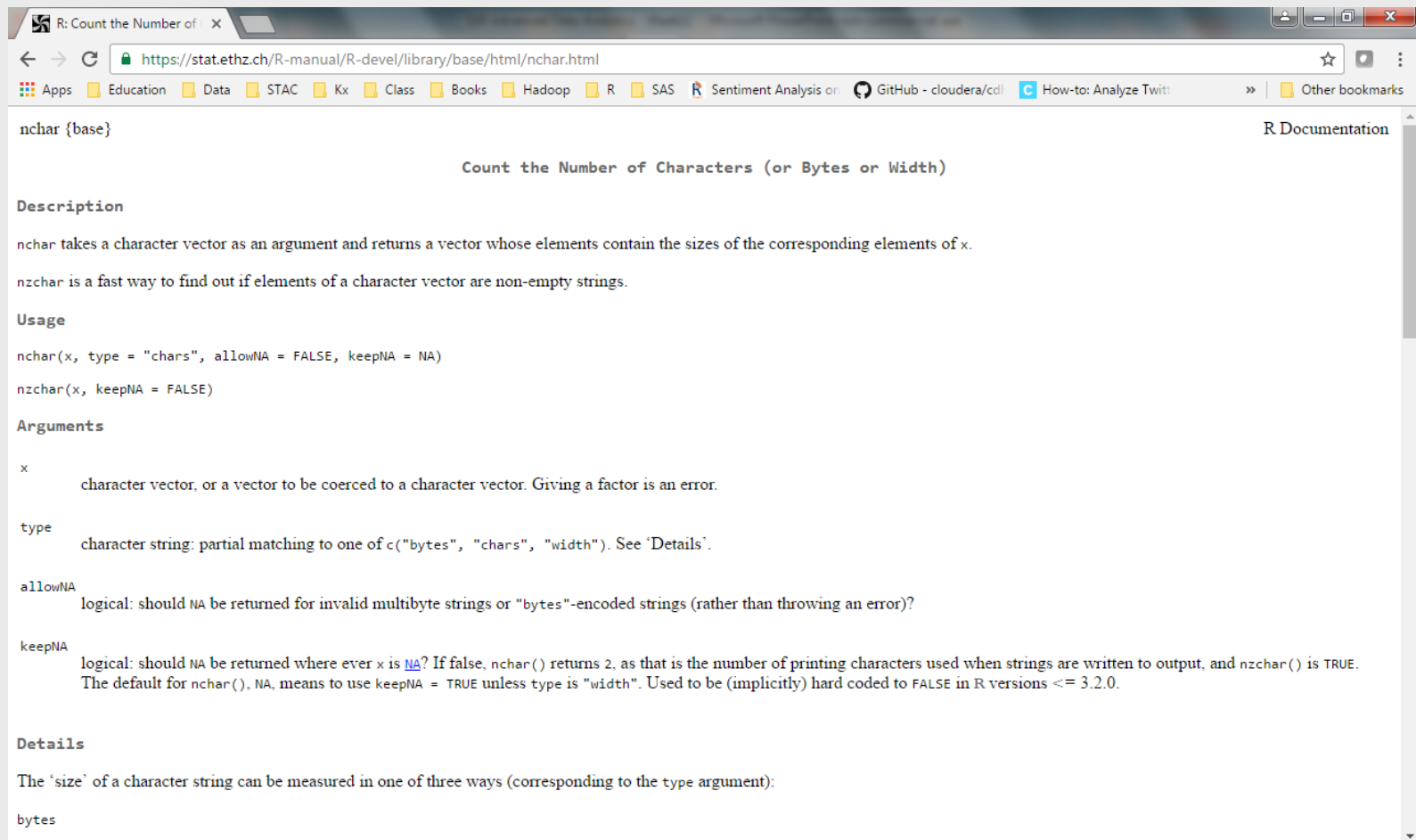
$v2
$v2[[1]]
[1] 2983

$v2[[2]]
[1] 1890
```

Week 11 Topic:

nchar

<https://stat.ethz.ch/R-manual/R-devel/library/base/html/nchar.html>



The screenshot shows a web browser window displaying the R Documentation page for the `nchar` function. The browser's address bar shows the URL `https://stat.ethz.ch/R-manual/R-devel/library/base/html/nchar.html`. The page title is "nchar {base}" and the subtitle is "Count the Number of Characters (or Bytes or Width)".

Description

`nchar` takes a character vector as an argument and returns a vector whose elements contain the sizes of the corresponding elements of `x`.

`nchar` is a fast way to find out if elements of a character vector are non-empty strings.

Usage

```
nchar(x, type = "chars", allowNA = FALSE, keepNA = NA)
nzchar(x, keepNA = FALSE)
```

Arguments

- `x`: character vector, or a vector to be coerced to a character vector. Giving a factor is an error.
- `type`: character string: partial matching to one of `c("bytes", "chars", "width")`. See 'Details'.
- `allowNA`: logical: should `NA` be returned for invalid multibyte strings or "bytes"-encoded strings (rather than throwing an error)?
- `keepNA`: logical: should `NA` be returned where ever `x` is `NA`? If false, `nchar()` returns 2, as that is the number of printing characters used when strings are written to output, and `nzchar()` is `TRUE`. The default for `nchar()`, `NA`, means to use `keepNA = TRUE` unless `type` is "width". Used to be (implicitly) hard coded to `FALSE` in R versions `<= 3.2.0`.

Details

The 'size' of a character string can be measured in one of three ways (corresponding to the `type` argument):

bytes

Week 11 Topic: order

http://www.cookbook-r.com/Manipulating_data/Sorting/#reverse-sort

The screenshot shows a web browser window with the address bar displaying `www.cookbook-r.com/Manipulating_data/Sorting/#reverse-sort`. The page title is "Reverse sort". The main content area contains the following text:

The overall order of the sort can be reversed with the argument `decreasing=TRUE`.

To reverse the direction of a particular column, the method depends on the data type:

- Numbers: put a `-` in front of the variable name, e.g. `df[order(-df$weight),]`.
- Factors: convert to integer and put a `-` in front of the variable name, e.g. `df[order(-xtfrm(df$size)),]`.
- Characters: there isn't a simple way to do this. One method is to convert to a factor first and then sort as above.

On the right side of the page, there is a sidebar with links: Problem, Solution, Vectors, Data frames, and Reverse sort.

```
# Reverse sort by weight column. These all have the same effect:
arrange(df, -weight) # Use arrange from plyr package
df[ order(df$weight, decreasing=TRUE), ] # Use built-in R functions
df[ order(-df$weight), ] # Use built-in R functions
#> id weight size
#> 2 2 27 large
#> 3 3 24 medium
#> 4 4 22 large
#> 1 1 20 small

# Sort by size (increasing), then by weight (decreasing)
arrange(df, size, -weight) # Use arrange from plyr package
df[ order(df$size, -df$weight), ] # Use built-in R functions
#> id weight size
#> 2 2 27 large
#> 4 4 22 large
#> 3 3 24 medium
#> 1 1 20 small

# Sort by size (decreasing), then by weight (increasing)
# The call to xtfrm() is needed for factors
arrange(df, -xtfrm(size), weight) # Use arrange from plyr package
df[ order(-xtfrm(df$size), df$weight), ] # Use built-in R functions
#> id weight size
#> 1 1 20 small
#> 3 3 24 medium
#> 4 4 22 large
#> 2 2 27 large
```

Week 11 Topic:

sub

<http://rfunction.com/archives/2354>



The screenshot shows a web browser window with the address bar displaying `rfunction.com/archives/2354`. The page title is `sub, gsub | R Function of`. The browser's bookmark bar includes links for Apps, Education, Data, STAC, Kx, Class, Books, Hadoop, R, SAS, Sentiment Analysis on GitHub - cloudera/cd, How-to: Analyze Twitter, and Other bookmarks.

sub, gsub

`sub(pattern, replacement, x)`
`gsub(pattern, replacement, x)`

Replace the first occurrence of a pattern with sub or replace all occurrences with gsub.

- `pattern` – A pattern to search for, which is assumed to be a regular expression. Use an additional argument `fixed=TRUE` to look for a pattern without using regular expressions.
- `replacement` – A character string to replace the occurrence (or occurrences for `gsub`) of pattern.
- `x` – A character vector to search for pattern. Each element will be searched separately.

Example. The simple example below does not make use of regular expressions.

[Download the Code](#)

```
>  
> x <- c("This is a sentence about axis",  
+       "A second pattern is also listed here")  
> sub("is", "XY", x)  
[1] "ThXY is a sentence about axis"  
[2] "A second pattern XY also listed here"  
> gsub("is", "XY", x)  
[1] "ThXY XY a sentence about axXY"  
[2] "A second pattern XY also lXYted here"  
>
```

Access Connections
HOME-732E-5
Status: Connected
Speed: 300.0 Mbps
IP address: 10.0.0.127
Intel(R) Centrino(R) Advanced-N 6250 AGN

Week 11 Topic: Scoring by sentence

<https://blog.exploratory.io/twitter-sentiment-analysis-scoring-by-sentence-b4d455de3560#.iun83j6cd>

you will get the result like below.

SHARE
20

```
mutate(sentiment = get_sentiment(text))
```

text	sentiment
1 I'm feeling so good!	0.5
2 I'm feeling good.	0.5774
3 I'm feeling much better.	0.9
4 I'm not feeling good.	-0.5
5 I'm not feeling so good	-0.4472

We can see negations and intensifiers in the sentences are influencing the result.

Now, let's try it out with real world data using [Exploratory Desktop](#). You can

Never miss a story from **learn dplyr**, when you sign up for Medium.
[Learn more](#)

GET UPDATES

Week 11 Topic: Using sentimentr

<https://github.com/trinker/sentimentr>

Twitter Sentiment Analysis x GitHub - trinker/sentimentr x

GitHub, Inc. [US] | <https://github.com/trinker/sentimentr>

Apps Education Data STAC Kx Class Books Hadoop R SAS Sentiment Analysis on GitHub - cloudera/cdl How-to: Analyze Twitt Other bookmarks

Personal Open source Business Explore Pricing Blog Support This repository Search Sign in Sign up

trinker / sentimentr Watch 6 Star 63 Fork 10

Code Issues 5 Pull requests 1 Projects 0 Wiki Pulse Graphs

No description or website provided.

178 commits 1 branch 3 releases 2 contributors

Branch: master New pull request Find file Clone or download

Tyler Rinker Changed license to MIT Latest commit a346073 on Aug 23

R	In highlight colons and semicolons were not parsed the same as the ma...	4 months ago
data	- added unprofessional to main olarity_dt	5 months ago
inst	upped version	3 months ago
man	In highlight colons and semicolons were not parsed the same as the ma...	4 months ago
tests	fixed test	6 months ago
.Rbuildignore	fixed image swap	6 months ago
.gitignore	improved default dictionaries	6 months ago

trinker-sentimentr-....zip Show all Show hidden icons

Week 11 Topic:

Contextual Valence Shifters

<https://www.aai.org/Papers/Symposia/Spring/2004/SS-04-07/SS04-07-020.pdf>

SS04-07-020.pdf 1 / 6

Contextual Valence Shifters

Livia Polanyi* and Annie Zaenen†

*FXPAL, 3400 Hillview Ave, Bldg. 4, Palo Alto CA 94304
 †PARC, 3333 Coyote Hill Road, Palo Alto, CA 94304 USA
 polanyi@fxpal.com, zaenen@parc.com

Abstract

In addition to describing facts and events, texts often communicate information about the attitude of the writer or various participants towards material being described. The most salient clues about attitude are provided by the lexical choice of the writer but, as discussed below, the organization of the text also contributes information relevant to assessing attitude. We argue that the current work in this area that concentrates mainly on the negative or positive attitude communicated by individual terms (Edmonds & Hirst 2002; Hatzivassiloglou and McKeown 1997; Turney & Littman 2002; Wiebe et al. 2001, In Press) is incomplete and often gives the wrong results when implemented directly. We then describe how the base attitudinal valence of a lexical item is modified by lexical and discourse context and propose a simple, “proof of concept” implementation for some contextual shifters.

From simple valence to contextually determined valence

Simple Lexical valence

Examples of lexical items that communicate a negative or positively attitude (*valence*) can be found in all open word classes and as multi-word collocations such as *freedom fighter*. Below we have listed some examples of English words which can be readily characterized as positively or negatively valenced¹.

PART OF SPEECH	Positive Valence	Negative Valence
Verbs	Boost, Embrace, Ensure, Encourage, Delight, Manage, Ease	Conspire, Meddle, Discourage, Fiddle, Fail, Haggle
Nouns	Approval, Benefit, Chance, Approval	Backlash, Backlog, Bankruptcy, Beat-

Introduction

SS04-07-020.pdf Show all