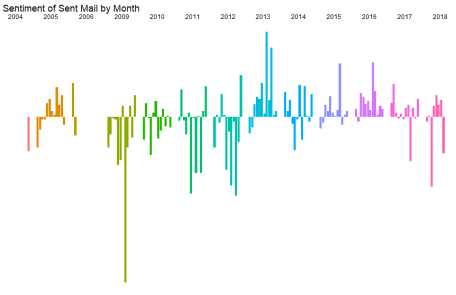
How can we use data analytics to increase our self-knowledge? Along with biofeedback from digital devices like [FitBit](https://dgarmat.github.io/Calories-vs-Sleep/), less structured sources such as sent emails can provide insights.

E.g. here it seems my communication took a sudden more positive turn in 2013. Let’s see what else shakes out of my sent email corpus.



In [Snakes in a Package: combining Python and R with reticulate](https://www.mango-solutions.com/blog/snakes-in-a-package-combining-python-and-r-with-reticulate) Adnan Fiaz uses a download of personal gmail from [Google Takeout](https://takeout.google.com/) to extract [R-bloggers](http://r-bloggers.com/) post counts from subject lines. To handle gmail’s choice of mbox file format, rather than write a new R package to parse mbox files, he uses [reticulate](https://rstudio.github.io/reticulate/articles/introduction.html) to import a Python package, [mailbox](https://docs.python.org/2/library/mailbox.html). His approach seems a great use case for reticulate – when you want to take advantage of a highly developed Python package in R.

**Loading Email Corpus into R**

I wanted to mine my own emails for sentiment and see if I can learn anything about myself. Has my sent mail showed signs of mood trends over time? I started by following his example:

library(tidyverse)

library(stringr)

library(tidytext)

library(lubridate)

library(reticulate)

mailbox <- import("mailbox")

sent <- mailbox$mbox("Sent-001.mbox")

message <- sent$get\_message(11L)

message$get("Date")

# [1] "Mon, 23 Jul 2018 20:01:33 -0700"

message$get("Subject")

# [1] "Re: Ptfc schedules"

Loading in email #11, can see it’s about Portland Football Club’s schedule. I wanted to see the body of the email, but found the normal built-in documentation doesn’t exist for Python modules

?get\_message

# No documentation for ‘get\_message’ in specified packages and libraries:

# you could try ‘??get\_message’

?mailbox

# No documentation for ‘mailbox’ in specified packages and libraries:

# you could try ‘??mailbox’

Returning message prints the whole thing, but with much additional unneeded formatting. So worked around it with nested sub() and gsub() commands on specific example emails to get down to the text I wrote and sent, only.

It starts with this already difficult to understand call

sub(".\*Content-Transfer-Encoding: quoted-printable", "",

gsub("=E2=80=99", "'",

gsub(">", "",

sub("On [A-Z][a-z]{2}.\*", "",

gsub("\n|\t", " ",

message)))))

And, after much guess-try-see-what’s-left-and-add-another-sub(), ended up with this ugly function that does semi-reasonably for my goal of sentiment analysis:

parse\_sent\_message <- function(email){

substr(

gsub("-top:|-bottom:|break-word","",

sub("Content-Type: application/pdf|Mime-Version: 1.0.\*","",

sub(".\*charset ISO|charset UTF-8|charset us-ascii","",

sub(".\*Content-Transfer-Encoding: 7bit", "",

sub("orwarded message.\*", "",

gsub("=|\"", " ",

gsub(" ", " ",

gsub("= ", "",

sub(".\*Content-Transfer-Encoding: quoted-printable", "",

sub(".\*charset=UTF-8", "",

gsub("=E2=80=99|'", "'",

gsub(">|<", "",

sub("On [A-Z][a-z]{2}.\*", "",

gsub("\n|\t||  
", " ",

email))))))))))))))),

1, 10000)

}

parse\_sent\_message(message)

# [1] " Hey aren't you planning to go to Seattle the 16th? Trying to figure out my days off schedule "

Good to go. I tried using the R [mailman](https://github.com/MangoTheCat/mailman) wrapper, but ran into issues, so went back to the imported mailbox module. Importing and parsing took a few minutes:

message$get("From") # check this email index 11 if from my email address

myemail <- message$get("From") # since it is, save as myemail to check the rest

keys <- sent$keys()

# keys <- keys[1:3000] # uncomment if want to run the below on a subset to see if it works

number\_of\_messages <- length(keys)

pb <- utils::txtProgressBar(max=number\_of\_messages)

sent\_messages <- data\_frame(sent\_date = as.character(NA), text = rep(as.character(NA), number\_of\_messages))

for(i in seq\_along(keys)){

message <- sent$get\_message(keys[i])

if(is.character(message$get("From"))){

if (message$get("From") %in% myemail){

sent\_messages[i, 1] <- message$get("Date")

sent\_messages[i, 2] <- parse\_sent\_message(message)

}

}

utils::setTxtProgressBar(pb, i)

}

If the message is not from me, it is saved as NA. What percent of mail flagged “sent” was not from myemail?

sum(is.na(sent\_messages$text)) / number\_of\_messages

# [1] 0.6664132

67%.  
Removing them and doing some additional processing, can see these 11,093 remaining sent emails range from November of 2014 to September of 2018 with a median date of October of 2013.

sent\_messages <-

sent\_messages %>%

filter(!is.na(text))

sent\_messages <-

sent\_messages %>%

mutate(sent\_date = dmy\_hms(sent\_date))

# remove duplicates per month

sent\_messages <-

sent\_messages %>%

mutate(year\_sent = year(sent\_date),

month\_sent = month(sent\_date)) %>%

group\_by(year\_sent, month\_sent, text) %>%

top\_n(1, wt = sent\_date) %>%

ungroup()

sent\_messages %>%

summary(sent\_date)

# sent\_date text year\_sent month\_sent

# Min. :2004-11-10 01:42:04 Length:11093 Min. :2004 Min. : 1.000

# 1st Qu.:2010-07-17 20:39:10 Class :character 1st Qu.:2010 1st Qu.: 3.000

# Median :2013-10-01 22:12:08 Mode :character Median :2013 Median : 6.000

# Mean :2013-03-24 10:55:30 Mean :2013 Mean : 6.416

# 3rd Qu.:2015-09-18 19:45:21 3rd Qu.:2015 3rd Qu.: 9.000

# Max. :2018-09-30 01:35:02 Max. :2018 Max. :12.000

While median date comes a bit later than the chronological midpoint seemingly implies slightly more emails later, from the chart above, it’s probably more due to missing years of data.

**Sentiment Analysis**

Julia Silge and David Robinson have put together an excellent online reference on text mining at [Text Mining with R](https://www.tidytextmining.com/) so with some slight work can follow their analyses with email data. Using their tidytext package, quickly see a lot of html formatting tags still made it past my gsub() gauntlet.

tidy\_emails <-

sent\_messages %>%

unnest\_tokens(word, text)

tidy\_emails

# # A tibble: 886,870 x 4

# sent\_date year\_sent month\_sent word

#

# 1 2018-09-27 16:30:19 2018 9 htmlbodyp

# 2 2018-09-27 16:30:19 2018 9 style

# 3 2018-09-27 16:30:19 2018 9 margin

# 4 2018-09-27 16:30:19 2018 9 0px

# 5 2018-09-27 16:30:19 2018 9 font

# 6 2018-09-27 16:30:19 2018 9 stretch

# 7 2018-09-27 16:30:19 2018 9 normal

# 8 2018-09-27 16:30:19 2018 9 font

# 9 2018-09-27 16:30:19 2018 9 size

# 10 2018-09-27 16:30:19 2018 9 12px

# # ... with 886,860 more rows

In fact, after common stop words are removed, can see a need to add a few more

data(stop\_words)

tidy\_emails <-

tidy\_emails %>%

anti\_join(stop\_words)

tidy\_emails %>%

count(word, sort = TRUE)

# # A tibble: 129,528 x 2

# word n

#

# 1 3d 8433

# 2 aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 7620

# 3 content 4086

# 4 dan 3487

# 5 1 3451

# 6 font 2735

# 7 type 2695

# 8 style 2535

# 9 nbsp 2495

# 10 class 2451

# # ... with 129,518 more rows

Maybe the

nchar("aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa")

# [1] 76

76 a’s in a row come from consolidating from something in the gsub()s.

Adding these less useful terms to create an email stop words dictonary:

email\_stop\_words <-

stop\_words %>%

rbind(

data\_frame("word" = c(seq(0,9), "3d", "8a", "mail.gmail.com", "wa", "aa", "content", "dir",

"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa",

"ad", "af", "font", "type", "auto", "zz", "ae", "zx", "id", "ai",

"style", "nbsp", "class", "span", "http", "text", "gmail.com",

"plain", "0px", "size", "color", "quot", "8859", "href", "margin", "ltr",

"left", "disposition", "attachment", "padding", "rgba", "webkit", "https"),

"lexicon" = "sent\_email")

)

# just remove all words less than 3 letters

tidy\_emails <-

tidy\_emails %>%

anti\_join(email\_stop\_words) %>%

filter(nchar(word) >= 3)

tidy\_emails %>%

count(word, sort = TRUE) %>%

top\_n(n = 10, wt = n) %>%

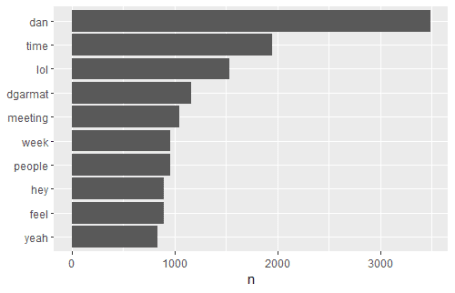
mutate(word = reorder(word, n)) %>%

ggplot(aes(word, n)) +

geom\_col() +

xlab(NULL) +

coord\_flip()



Can see some unsurprising name related common terms as well as “lol” and “hey”. But surprisingly “time”, “meeting”, “week”, and “people” also show up a lot. Wonder if those are unusual. (Would need another sent mail corpus to compare.)

What are my top joy words in email?

nrc\_joy <-

get\_sentiments("nrc") %>%

filter(sentiment == "joy")

tidy\_emails %>%

inner\_join(nrc\_joy) %>%

count(word, sort = TRUE)

# # A tibble: 373 x 2

# word n

#

# 1 art 531

# 2 feeling 389

# 3 hope 387

# 4 found 318

# 5 pretty 286

# 6 true 267

# 7 pay 229

# 8 money 218

# 9 friend 209

# 10 love 203

# # ... with 363 more rows

Hm, I only partially agree with this list. “Art” is a friend I email frequently. “Feeling” is a slight positive, but more neutral than a joy word per se. “Hope” is most common I’d agree with between 2004 and 2018 it seems.

How does sentiment look over time? Grouping by month:

email\_sentiment <-

tidy\_emails %>%

mutate(year\_sent = year(sent\_date),

month\_sent = month(sent\_date)) %>%

inner\_join(get\_sentiments("bing")) %>%

count(year\_sent, month\_sent, sentiment) %>%

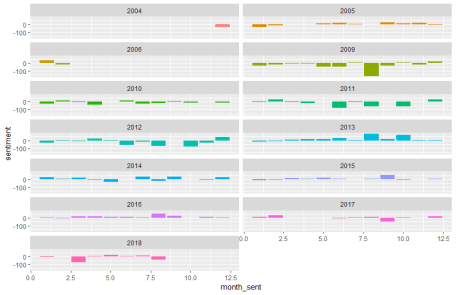
spread(sentiment, n, fill = 0) %>%

mutate(sentiment = positive - negative)

ggplot(email\_sentiment, aes(month\_sent, sentiment, fill = as.factor(year\_sent))) +

geom\_col(show.legend = FALSE) +

facet\_wrap(~year\_sent, ncol = 2)



2005, 2013, 2015 and 2016 look like more positive sentiment sent mail years. 2009 and 2011 look more negative overall. A few years, much of 2006, 2007 and 2008 are missing, weirdly.

Also see an apparently highly negative month in August of 2009.

# whoa happened in August of 2009?

sent\_messages %>% filter(sent\_date >= "2009-08-01", sent\_date <= "2009-08-31") %>% write.csv("temp.csv")

tidy\_emails %>%

mutate(year\_sent = year(sent\_date),

month\_sent = month(sent\_date)) %>%

inner\_join(get\_sentiments("bing")) %>%

filter(year\_sent == 2009, month\_sent == 08) %>%

count(word, sentiment, sort = TRUE)

# # A tibble: 237 x 3

# word sentiment n

#

# 1 pain negative 35

# 2 happiness positive 21

# 3 sting negative 21

# 4 happy positive 12

# 5 stinging negative 12

# 6 depression negative 11

# 7 free positive 11

# 8 bad negative 9

# 9 damage negative 9

# 10 venom negative 9

# # ... with 227 more rows

Was it a bad breakup? Digging into my emails, can find a New York Times Magazine article copy-and-pasted and sent to several people. The article, “Oh, Sting, Where Is Thy Death?” By Richard Conniff, mentions the pain of stinging insects and its relevance to happiness research. Note most of those ns are divisible by 3.

**Most Common Charged Words**

If taking all the emotionally charged words and seeing what comes out most often, both surprises and expected outcomes show up:

bing\_word\_counts <-

tidy\_emails %>%

inner\_join(get\_sentiments("bing")) %>%

count(word, sentiment, sort = TRUE) %>%

ungroup()

bing\_word\_counts

# # A tibble: 2,143 x 3

# word sentiment n

#

# 1 cool positive 481

# 2 nice positive 456

# 3 free positive 445

# 4 bad negative 308

# 5 pretty positive 286

# 6 retreat negative 239

# 7 solid positive 230

# 8 fine positive 222

# 9 hard negative 219

# 10 worth positive 207

# # ... with 2,133 more rows

bing\_word\_counts %>%

group\_by(sentiment) %>%

top\_n(10) %>%

ungroup() %>%

mutate(word = reorder(word, n)) %>%

ggplot(aes(word, n, fill = sentiment)) +

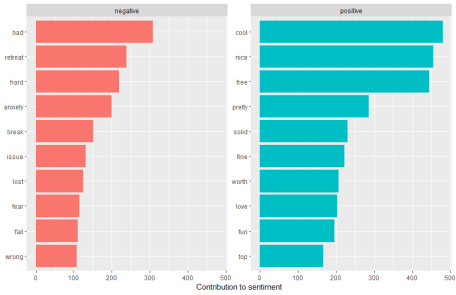
geom\_col(show.legend = FALSE) +

facet\_wrap(~sentiment, scales = "free\_y") +

labs(y = "Contribution to sentiment",

x = NULL) +

coord\_flip()



Surprised to see how much more positive words show up than negative words – Bing does have more positive words in its lexicon, so could make sense there. “Bad” as top negative word seems like a bad top word. “Issue” is definitely a word I have an issue with using a bad amount of time. But it’s cool to see how much I use “cool” (or is it bad? this is causing anxiety). Anyway, I think this is a solid view worth the time to get a nice feeling for top words I love to use in email.

**Obligatory Wordcloud**

Is it easier to read than the above? Nah, but it must be included in any text mining blog post, so…

library(wordcloud)

tidy\_emails %>%

anti\_join(email\_stop\_words) %>%

filter(nchar(word) >= 3) %>%

count(word) %>%

with(wordcloud(word, n, max.words = 100))

library(reshape2)

tidy\_emails %>%

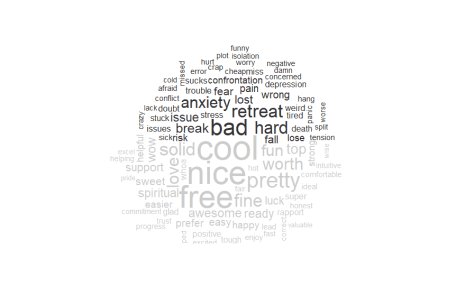
inner\_join(get\_sentiments("bing")) %>%

count(word, sentiment, sort = TRUE) %>%

acast(word ~ sentiment, value.var = "n", fill = 0) %>%

comparison.cloud(colors = c("gray20", "gray80"),

max.words = 100)



Hope that was cool