In this post, we’ll take a look at a basic text visualization technique: word clouds. However, part of my job is providing data analysis and visualizations for senior management, and the basic word cloud approaches one finds straight out-of-the box don’t easily accommodate this use case. Below, I’ll outline my workflow for making customizable word clouds that you won’t be afraid to show to anyone in your organization!

**Data Visualization for Management Presentations**

It’s not easy to bridge the gap between analytics and management, and to ensure that data analysis is properly communicated to business stakeholders. This communication is an essential part of the job, and without it, the chances that your data analysis will have any impact are very small.

A critical aspect of communicating any type of data analysis is story telling, e.g. telling a narrative that relies on data analysis and suggests actionable conclusions. When presenting to management stakeholders, it’s very important to tell a story that flows logically and unimpeded towards its conclusion. Such presentations can easily get derailed when they are not focused, are structured based on methodological considerations (e.g. telling the story of the data analysis, not of the business problem + relevant insight), or present visuals that are unclear or invite excess scrutiny. Essentially, when presenting the results of a data analysis to management, you don’t want anything to distract from the data-driven conclusions you’re trying to convey.

**Problems with Out-of-the-Box Word Clouds**

The need to present clear and intuitive data visualizations is therefore of paramount importance. However, when using out-of-the-box word cloud routines in R, I’ve noticed two primary issues that make it difficult to make compelling visualizations for management stakeholders.

1. [Stemming](https://en.wikipedia.org/wiki/Stemming) (removing the end of a word to harmonize different forms, e.g. *argue*, *argued*, *argues*, *arguing* are all truncated to *argu*) is essential to getting good word counts, but stemmed words in word clouds look strange and are therefore distracting (e.g. they can easily derail a presentation into methodological discussions about text processing, rather than the implications of your data analysis for decision-making).
   * However, to the best of my knowledge, none of the the packages for text analysis or word clouds in R make it easy to “un-stem” a word. From a strictly data analytic point of view, you would never really want to do this. For the current use case, though, it’s essential to do so.
2. Sometimes you want to remove a word from a word cloud, even if it occurs very frequently. For example, the name of your company could occur very frequently (in internal documents or open-ended responses on surveys), but it doesn’t tell you anything insightful about the topic under study. In such cases, these words are often prominent in the word cloud (because they are used quite often), but have no added-value, don’t help you advance your story, and at worst can distract from other important topics that appear in the word cloud.
   * While it is possible in every text analysis package to remove words from a document, corpus, or document-term frequency matrix, this typically occurs far upstream from the code used to make the word cloud. As such, it is not always straightforward to do so, particularly if you have many different columns in your dataset that should be turned into word clouds.

**My Workflow for Word Clouds for Management Presentations**

In this post, we’ll go through a work flow that I use in order to remedy the two above-mentioned problems with existing word cloud packages. There’s lots of great things about this package, but something I really appreciate is that the package developers have thought a lot about making common text analytic procedures (e.g. stemming, term weighting, n-gram selection, removing numbers, etc.) very robust and easy-to-use.

The work flow uses a number of custom-built functions, which we’ll go over below. There are separate functions for all of the different steps that we need in the “Quanteda way” of analyzing text data. We first turn the text field in our dataframe into a corpus, from which we extract and clean text tokens (e.g. terms or words). We then convert the tokens into a document-feature matrix, and pass this along to the Quanteda wordcloud routine to create word clouds. Built into this workflow, I’ve created ways to specify words that should be replaced and their replacements (effectively “un-stemming” stemmed words) and to specify words which should be removed from the word cloud. Once the functions have been defined, it’s very easy to make a basic out-of-the box word cloud, examine it to see what needs to be changed, and to then re-make the word cloud with these changes taken into account.

**Step 1: Data Frame and Text Field to Corpus**

The first function takes a data frame with a text field and creates a corpus object. The **corpus** is the most basic element in the Quanteda text process flow, and is essentially a “library” of original documents which are stored along with meta-data at the corpus level and at the document-level.

# Function 1: generate a corpus from the text field in original data

# input: data frame + text field

# output: corpus (Quanteda basic object)

comment\_field\_to\_corpus <- function(data\_f, text\_field\_f){

corpus\_f <- corpus(data\_f, text\_field = text\_field\_f)

# return the corpus object

return(corpus\_f)

}

**Step 2: Corpus to Cleaned Tokens Object**

The second function takes the corpus object we created with the first function, performs a number of text cleaning operations, and returns a tokens object (a list of *tokens* in the form of character vectors, where each element of the list corresponds to an input document).

Specifically, we remove punctuation, numbers, and symbols. We then convert all the letters to lower case, and stem the words (e.g. removing the end of the word to harmonize different variations on the same root). Finally, we remove any remaining words that are less than 3 characters and select unigrams and bigrams (e.g. 1 and 2 word combinations). Finally, the function returns the cleaned tokens object.

# Function 2: generate cleaned tokens from the corpus

# input: corpus

# output: cleaned tokens object

make\_clean\_tokens <- function(corpus\_f){

# make the base tokens object

# remove punctuation, numbers, symbols

# and stopwords (note: default stopword matching is case-insensitive)

clean\_toks\_f <- tokens(corpus\_f, what = 'word', remove\_punct = TRUE,

remove\_numbers = TRUE, remove\_symbols = TRUE, verbose = TRUE)

clean\_toks\_f <- tokens\_remove(clean\_toks\_f,

stopwords("english"))

# convert all letters to lower case

clean\_toks\_f <- tokens\_tolower(clean\_toks\_f)

# then stem

clean\_toks\_f <- tokens\_wordstem(clean\_toks\_f, language = quanteda\_options("language\_stemmer"))

# remove words less than three characters

clean\_toks\_f <- tokens\_select(clean\_toks\_f, selection = "keep", min\_nchar = 3)

# select bigrams and unigrams

clean\_toks\_f <- tokens\_ngrams(clean\_toks\_f, n = 1:2)

# return the cleaned tokens object

return(clean\_toks\_f)

}

**Step 3: Cleaned Tokens to DFM (Document-Feature Matrix)**

The third function takes the cleaned tokens and generates a DFM, which is a matrix associating values for certain features with each document, with the documents in the rows and “features” in the columns.

# Function 3: generate dfm (document feature matrix) from cleaned tokens

# input: cleaned tokens object

# output: document-feature matrix (input for word clouds)

# there are two options for constructing the dfm:

# 1) term frequency (e.g. counts of the words across all documents)

# 2) document frequency (e.g. count of the number of documents

# containing each word)

make\_dfm <- function(tokens\_f, dfm\_method\_f){

# make a dfm object from the tokens

dfm\_f <- dfm(tokens\_f, verbose = TRUE)

# two possible methods implemented: term\_freq and doc\_freq

# if we choose the "term\_freq" method:

if(dfm\_method\_f == 'term\_freq'){

# return the DFM object

return(dfm\_f)

# if we choose the "doc\_freq" method:

} else if(dfm\_method\_f == 'doc\_freq'){

# make a boolean (e.g. 0/1) weighting

# this means that the document

# frequency to build the wordcloud

dfm\_f <- dfm\_weight(dfm\_f, scheme = "boolean")

# return the DFM object

return(dfm\_f)

} else{

stop("invalid method specified for creating the dfm. options are 'term\_freq' and 'doc\_freq' ")

}

}

**Step 4: Remove Words We Don’t Want in the Word Clouds**

The fourth function removes the words that we do not want to see in the word clouds. For example, when mining internal documents, the name of one’s company might occur very frequently. But this is rarely interesting or informative in the larger context of the data analysis, and provides no useful insight upon which to make a decision. Such words can only distract from the main point of the presentation, and so it’s a good idea to remove them.

In the function below, we remove the words from the tokens object created in Step 2 above.

# Function 4: Remove Words

# this functions removes words that we do not

# want to see in our wordclouds

# it removes the words from the tokens object

# before it is turned into a dfm

remove\_words <- function(tokens\_f, words\_to\_remove\_f){

# use "tokens select" to remove the selected words

# from the tokens object

trimmed\_tokens\_f <- tokens\_select(tokens\_f,

words\_to\_remove\_f ,

selection = "remove",

case\_insensitive = TRUE)

# return the tokens object, with the chosen words removed

return(trimmed\_tokens\_f)

}

**Step 5: “Un-stem” the Stemmed Words**

The fifth function allows us to specify a list of “to-be-replaced” words (e.g. “busi”) and the “replacement” words (e.g. “business”). Using this method, we can ensure that we don’t have any truncated words in our word cloud.

# Function 5: Clean Stemmed Words

# this function "un-stems" the stemmed words

# we pass a vector of the words-to-replace (e.g. "busi")

# and the replacements (e.g. "business")

# this function operates on the dfm object

clean\_stemmed\_words <- function(dfm\_f, old\_words\_f, new\_words\_f){

#

#

# create a combination of renamed and original feature names

names(new\_words\_f) <- old\_words\_f

# assign the replacements to the dfm

colnames(dfm\_f) <- str\_replace\_all(colnames(dfm\_f), new\_words\_f)

# compress the dfm

dfm\_f <- dfm\_compress(dfm\_f)

# and return the cleaned dfm object

return(dfm\_f)

}

**Step 6: Master Cleaning Function**

The sixth and final function puts together all of the component functions we have defined above. The function takes a data frame and a text field, along with some optional parameters (e.g. words-to-replace, words to remove), and returns a dfm that is cleaned according to our specifications. We can pass this dfm directly to the Quanteda word cloud plot method to make our word cloud.

# Function 6: master cleaning function

# this puts together all of the pieces we have defined above

# the input is the dataframe + text column

# (along with optional paramaters, e.g. words to remove)

# the output is a DFM object we can pass directly to the

# Quanteda wordcloud function

master\_cleaning\_function <- function(data\_f,

text\_field\_f,

dfm\_method\_f,

old\_words\_f,

new\_words\_f,

words\_to\_remove\_f){

# create the corpus object using function defined above

corpus\_master\_f <- comment\_field\_to\_corpus(data\_f, text\_field\_f)

# create the cleaned tokens object using function defined above

clean\_toks\_master\_f <- make\_clean\_tokens(corpus\_master\_f)

# if we don't specify that we want to remove any words,

# we create the dfm directly from the tokens object

if(missing(words\_to\_remove\_f)){

dfm\_f <- make\_dfm(clean\_toks\_master\_f, dfm\_method\_f)

} else {

# if we want to remove words, we remove them from the

# tokens object using the function defined above

clean\_toks\_master\_f <- remove\_words(clean\_toks\_master\_f, words\_to\_remove\_f)

# and then create the dfm using the function defined above

dfm\_f <- make\_dfm(clean\_toks\_master\_f, dfm\_method\_f)

}

# if we don't specify words-to-be-replaced and their replacements

# we return the dfm directly

if(missing(old\_words\_f) & missing(new\_words\_f)) {

return(dfm\_f)

} else {

# if we specify words-to-be-replaced and replacements

# we make the changes to the dfm using the function

# defined above

dfm\_f <- clean\_stemmed\_words(dfm\_f, old\_words\_f, new\_words\_f)

# and then return the modified dfm

return(dfm\_f)

}

}

**A Worked Example: Word Clouds for Management with Wine Data**

As an illustrative example, the first text in the data set looks like this:

"Our 2007 Traditions Merlot features grapes from 12 different sites in the Columbia Valley. The diversity of terroir allows us to create a more balanced and complete wine. We also included small amounts of Cabernet Sauvignon, Malbec and Petite Sirah in the final blend for supplemental aromatics, texture and complexity. The nose combines varietal and regional signatures. It delivers forward fruit flavors of ripe plums, cherries, blueberries, mocha, vanilla and caramel. The palate is plush, soft and generous. This is an elegant, balanced, supple Merlot that pairs well with salmon, smoked meats, BBQ, game, ham, stew and cheeses."

In what follows, we will assume you have already defined the above functions in your R session.

**Part 1: “Out-of-the-Box” Word Clouds**

We first make an “out-of-the-box” word cloud, displaying the words “as is”, given the cleaning functions used in text processing. This visualization will allow us to see which specific words in our corpus should be changed or removed. (The changes we often want to make are unique to each context and data set, so it’s not possible to automate them.)

In the code below, we first define the color palette we will use in the word cloud. We then pass the data frame and the text field to our master function, without specifying any changes. In this example, we will work with the term frequencies (e.g. sum of word usage across all the documents). The code has an example using document frequencies.

# load the packages we'll need

library(quanteda)

library(plyr); library(dplyr)

library(stringr)

library(RColorBrewer)

# color parameters for plots

pal=brewer.pal(8,"Blues")

pal=pal[-(1:3)]

colors=brewer.pal(8, "Dark2")

# make the dfm using term frequencies

tf\_dfm <- master\_cleaning\_function(data\_f = raw\_data,

text\_field\_f = "Winemakers\_Notes",

dfm\_method\_f = 'term\_freq')

# plot the word cloud using the Quanteda word cloud function

quanteda::textplot\_wordcloud(tf\_dfm, max\_words = 25, col = colors,

min\_size = .5, max\_size = 7, rotation = 0)

Which returns the following plot:

**Part 2: Specifying Changes and Making the Final Word Cloud**

This already looks very nice! There are, however, a couple of changes needed to make a “management-ready” word cloud.

First, the most prominent word is “wine.” This makes perfect sense – these texts describe wine. However, we already knew that, and having the term dominate the plot does not add any insight or value. Let’s remove this word!

Second, we notice some stemmed words in the plot. For example, “palate” is truncated to “palat.” In the code below, I specify each of the stemmed words (*old words* in the function below) and indicate which words should serve as replacements (*new words* in the function below).

We define our modifications and pass everything to our master function like so:

# specify modifications to the words

old\_words <- c('palat', 'intens', 'miner', 'cherri', 'balanc')

new\_words <- c('palate', 'intense', 'mineral', 'cherry', 'balance')

to\_remove <- c('wine')

# make the cleaned dfm

tf\_dfm\_clean <- master\_cleaning\_function(data\_f = raw\_data,

text\_field\_f = "Winemakers\_Notes",

dfm\_method\_f = 'term\_freq',

old\_words\_f = old\_words,

new\_words\_f = new\_words,

words\_to\_remove\_f = to\_remove)

# make the word cloud with the changes specified above

quanteda::textplot\_wordcloud(tf\_dfm\_clean, max\_words = 25, col = colors,

min\_size = .5, max\_size = 7, rotation = 0)

Which returns the following plot:

This looks perfect! We have removed the term “wine” which didn’t add anything, and we’ve “un-stemmed” all of the stemmed words. There’s nothing here that will be an obvious distraction if we present this visualization to management!

**Summary and Conclusion**

In this post, we focused on text analysis and data visualization. We outlined a process to create customizable word clouds for use in management presentations, removing frequent words that did not add any insight and ensuring that all words in the visualization were complete, even after stemming.

This process allows us to produce “management-ready” visuals for presentation to decision-makers. I recently had to prepare a number of word clouds for presentation to senior management, and in every case, it was necessary to make these types of tweaks to the word clouds. The workflow described in this post made the task very straightforward and saved a tremendous amount of time when conducting the analyses. Based on how the presentations were received, I would also say that effort spent to customize the word clouds was definitely worth it!

