Assignment 3: Emoji Zipfian

In this assignment you will use regular expressions to count Emoji occurences and plot their distribution.

1 New Emoji list

I have scraped a new Emoji list from a different source that is somewhat more suitable for our purposes. Please download **here** and store it in your project folder.

2 Count Emojis

2.1 Identify Emojis

Load `data_stream` and (new) `emoji_ids`. Iterate over Emojis and test whether Emoji is in text. Use, e.g., `grepl()` on collapsed text of tweets (`paste(text, collapse = ' ')`).

Hint: Use the column containing the Emojis not the Unicode string.

```r
# load data_stream and emojis
data_stream = read.csv('data_stream.csv')
emoji_ids = read.table('proper_emoji_ids.txt',sep='\t',header=T)

# combine data stream
txt = paste(unlist(data_stream$text),collapse = ' ')

# check if emojis are in text
in_text = c()
for(i in 1:length(emoji_ids$emoji)){
  in_text[i] = grepl(emoji_ids$emoji[i],txt)
  }
```

2.2 Count Emojis

Iterate over identified Emojis and count how often each of them occurs. Use, e.g., `gregexpr()`.

```r
# select identified emojis
identified = emoji_ids$emoji[in_text]

# count emojis
cnts = c()
for(i in 1:length(identified)){
  pos = gregexpr(identified[i],txt)[[1]]
  if(pos[1] == -1){
    cnts[i] = 0
    } else {
    cnts[i] = length(pos)
    }
  }
```

3 Expand analysis

New streams

Now that you knwo the drill. Rerun your code of Assignment 1 for *three* different track terms for *five* minutes (aka 300s) each.

Store data streams in your project folder.

```r
# you know how the streaming works

# get streams and extract collapsed texts
texts = c()
track = c('YouTube','Trump','Brexit')
for(i in 1:length(track)){
  stream = readRDS(paste0('Streams/',gsub(' ','_',track[i]),'_stream.RDS'))
  texts[i] = paste(stream$text, collapse = ' ')
  }
```

Count Emojis

Then rerun analysis of section 2 (Count Emojis) for each of the three new data streams.

Store counts in your project folder.

```r
# extract emojis
emoji_ids = readRDS('advancedEmojiList.RDS')
utf8 = emoji_ids$utf8
natv = emoji_ids$native_mac

# create logical vector
cnts = matrix(NA,ncol=3,nrow=length(utf8))

# iterate over emojis
for(i in 1:length(natv)){

  # iterate over streams
  found = c()
  for(j in 1:length(texts)){

    # find locations
    pos1 = gregexpr(utf8[i], texts[j])[[1]]
    pos2 = gregexpr(natv[i], texts[j])[[1]]
    pos = c(pos1,pos2)
    pos = pos[pos!=-1]

    # store result
    cnts[i,j] = length(pos)

    }
  }
```

4 Plot

Plot the three sets of Emoji counts (y-axis) for the three data streams against their Unicode strings (x-axis) from largest to smallest count.

Choose your adventure

Novice Plot one set of counts using high level `plot()` function. Decide whether to plot points or lines using the `type` argument. Control axis labels using `xlab` and `ylab`. Control size of points, labels, and axes through various `cex` arguments. Make sure `xlim` and `ylim` fit the other sets of counts. Add other sets of counts using `points()` or `lines()`. Distinguish the three sets using different `pch` or `lty`, respectively. Add legend using `legend()`.

Expert:base Setup a canvas using `plot.new()` and `plot.window`. Plot sets of counts using different `points()` or `lines()`. Add labels and axes using `mtext()`. Add legend using `legend()` or by hand using, e.g., `text()`, `lines`, `points`, and `rect`.

Expert:ggplot Figure it out. Very powerful, but not my cup of tea.

Add Zipf's Law to plot

Add additional line or points representing Zipf's law. Specifically, plot

$$f(rank) = \frac{1}{(rank + \beta)^{\alpha}}$$

using $\alpha \approx 1$ and $\beta \approx 2.7$.

To do this define function that returns $f(rank)$ as a function of $rank$, α, and $beta$. Then create a sequence from 1 to the number of Emojis in the plot and compute $f(rank)$. Add result to the plot.

Store plot using `png()`

Use `png()` to store your plot on the harddrive. To do this execute `png()` before executing your plot code. Feed it with a `filename` (with proper extension, i.e., .png) and dimensions (`height` and `width`).

After executing the device function (`png()`), and your plot code, execute `dev.off()` to finalize the plot.

Publish plot

Post your final plot on *twitter* using *#nlpbasel*.

Bonus level

Plot Emojis as x-axis labels. Download Emoji images **here**. Use function below to add Emoji png images.

Install and load `png` package. Identify Unicode strings of the to be plotted Emojis. Use `list.files()` to retrieve all Emoji filenames (Note: `full.names = TRUE` returns full path). Select filenames of to be plotted Emojis using regular expressions.

Hint: Emoji's can only be plotted into `plot region` (see `mai` in `?par`). Hint: If you used ggplot, check out **emoGG**.

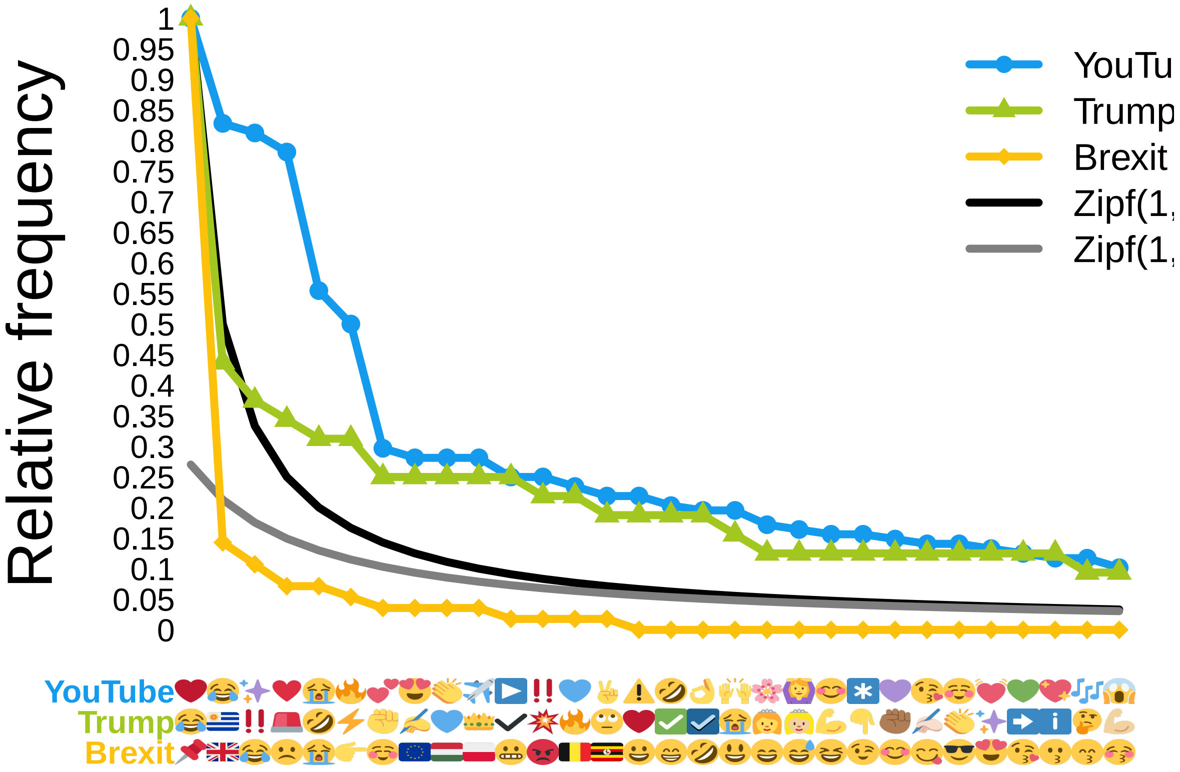
```r
require(png)
require(yarrr)
```

```
## Warning: package 'coda' was built under R version 3.3.2
```

```
## Warning: package 'Matrix' was built under R version 3.3.2
```

```r
# ------ Preprocess counts

# add code point labels
emj_cnts = data.frame(emoji_ids$code_point, cnts)
names(emj_cnts) = c('code_point',track)
```

```r
# split, normalize, and order
cnts_list = list()
for(i in 1:ncol(cnts)){

  # extract counts
  cnt = emj_cnts[,c(1,i+1)]

  # normalize
  cnt[,2] = cnt[,2] / max(cnt[,2])

  # order
  cnt = cnt[order(cnt[,2],decreasing = T),]

  # store
  cnts_list[[i]] = cnt
  }

# ------ Helpers

# zipfian
zipfian = function(rank, alpha, beta) 1 / ((rank + beta)**alpha)

# add emojis
add_emoji = function(filename, x, y, cex, match = F){
  pic  = readPNG(filename)
  dims = dim(pic)[1:2]
  usr = par()$usr
  if(match == T) ar = diff(usr[3:4]) / diff(usr[1:2]) else ar = 1
  rasterImage(pic, x-cex/2, y-(ar*cex/2), x+cex/2, y+(ar*cex/2), interpolate=TRUE)
  }

# ------ Plot

#png('EmojiZipfian.png',width = 720, height = 500)

xlim = c(.5,30.5)
ylim = c(-.25,1)
cols = piratepal("basel")[6:8]
pos = c(-.1,-.15,-.2)
pchs = c(16, 17, 18)
plot.new()
par(mar = c(0,4.5,1,1))
plot.window(xlim = xlim, ylim = ylim)
mtext(seq(0,1,.05),at=seq(0,1,.05),las=1,side=2, line=-1)
mtext(track,at = pos, line = -1, side = 2, las = 1, font = 2, col = cols)
mtext('Relative frequency',side=2, line = 2, cex = 2, at = .5)
legend(25,1, legend = c(track, 'Zipf(1,0)', 'Zipf(1,2.7)'),
       col = c(cols,'black','grey50'), lwd = 4, lty = 1, cex = 1.2,
       bty = 'n', pch = c(pchs, NA, NA))
lines(zipfian(1:30, 1, 0), lwd = 4, lty = 1, col = 'black')
lines(zipfian(1:30, 1, 2.7), lwd = 4, lty = 1, col = 'grey50')
for(i in 1:3){
  tmp = cnts_list[[i]][1:30,]
  lines(tmp[,2],lty = 1, lwd = 4, col = cols[i])
  points(tmp[,2],lty = 1, pch = pchs[i], col = cols[i], cex = 1.3)
  for(j in 1:30){
    path = paste0('emoji_imgs/',tmp[j,1],'.png')
    if(file.exists(path)){
      add_emoji(path,j,pos[i],1,match=T)
      } else {
      add_emoji('NAicon.png',j,pos[i],1,match=T)
      }
    }
  }
```



```
#dev.off()
```

End