```
# This might become an appendix
# R and GNUplot
# Handy Commands
# Save()
# which(keysAndPureRefs == "@justinbieber")
# points() and lines() adds to a graph
# axis(1, at=1:10, labels = sort1.keysAndPureRefs[1:10,1])
# keysAndPureRefs = data.frame(keys[,1], as.numeric(tweetRefPure[,1]))
/* sort1.keysAndPureRefs = keysAndPureRefs[order(keysAndPureRefs[,2], decreasing =
TRUE), ] */
# sort -t$'\t' -k3 -nr /home/melvin/sorted_results2_bthese > sorted_results2_bthese
# /home/melvin/final_results_bthese
# savePlot(filename = "plot5.png")
# which(keys==1dtorontooo)
# in R an array index works like this: [row , column]

# What is the total amount of tweets with @mentions compared to the total amount of smilies &
# @mentions? This could indicate the usefulness of this method.

# All characters that are numeric are converted to a numeric type.
#U:\\final_results_bthese
#/home/melvin/final_results_bthese
results = read.table(file="/home/melvin/final_results_bthese",
col.names=c("KEYS", "#TWEETS", "#TWEETS AND SMILIES", "#POS", "#NEG"), header =
TRUE, sep="\t")

# Make it more human like
keys = results[1]
tweetRefPure = results[2]
tweetRefHasEmoticons = results[3]
positiveEmoticons = results[4]
negativeEmoticons = results[5]

# Ratio between positive and negative
totalEmoticons = sum(positiveEmoticons + negativeEmoticons)
ratioOfEmoticons = sum(positiveEmoticons) / totalEmoticons # biased towards
positiveEmoticons
# ratioOfEmoticons = 0.8341719

sum(totalEmoticons)
# sum(totalEmoticons) = 930343

totalAtMentions = sum(tweetRefPure + tweetRefHasEmoticons)
ratioOfAtMentions = (sum(tweetRefPure) / totalAtMentions) # biased towards tweetRefPure
ratioOfAtMentions
# ratioOfAtMentions = 0.7719932 # note this only increases over time

# Converting to matrix (aka array)
tweetRefPure = as.matrix(tweetRefPure)
tweetRefPure = sort(tweetRefPure, decreasing=TRUE)
```

```
# Doing the same for tweetRefHasEmoticons
tweetRefHasEmoticons = as.matrix(tweetRefHasEmoticons)
tweetRefHasEmoticons = sort(tweetRefHasEmoticons, decreasing=TRUE)

# Plotting both at the same time
plot(tweetRefPure, log="xy" ,col="red", ylab="Frequency of being mentioned", xlab="Number of
data points", main="@mentions of users, with and without emoticons")
points(tweetRefHasEmoticons, col="green")

# Taking the special few...
ratioRef = tweetRefHasEmoticons / (tweetRefPure + tweetRefHasEmoticons)
ratioRef = as.matrix(ratioRef)
ratioRef = sort(ratioRef, decreasing=TRUE)
# Emoticons are used to some people a lot, there is this whole elite group
plot(ratioRef)

# converting everything to the matrix
tweetRefHasEmoticons = as.matrix(tweetRefHasEmoticons)
tweetRefPure = as.matrix(tweetRefPure)
positiveEmoticons = as.matrix(positiveEmoticons)
negativeEmoticons = as.matrix(negativeEmoticons)

# Plotting fun stuff
plot(tweetRefPure, tweetRefHasEmoticons, xlim=c(0,100), ylim=c(0,100))
plot(positiveEmoticons, negativeEmoticons, xlim=c(0,100), ylim=c(0,100))
plot(tweetRefHasEmoticons,positiveEmoticons, xlim=c(0,100), ylim=c(0,100)) /* strong
correlation */
plot(tweetRefHasEmoticons,negativeEmoticons, xlim=c(0,100), ylim=c(0,100)) /* less strong
correlation */

plot(tweetRefPure,positiveEmoticons, xlim=c(0,100), ylim=c(0,100))

# Searching for outliers
ratio = positiveEmoticons / (positiveEmoticons + negativeEmoticons)
array = c()
for (i in 1:length(ratio[,1])){
        temp = as.numeric(ratio[i,])
        if(temp > 0.98 && temp<1){
                array = c(array, temp)
        }
}
#results[which(array[1]==ratio),]

#@mentions vs positive emoticons
plot(totalRef[,1],positiveEmoticons[,1], xlim=c(1,35000),ylim=c(1,35000), log="xy", xlab="Total
References Frequency", ylab="Positive Emoticons Frequency", main="All @mentions vs
Positive Emoticons")

#l33t @mentions vs positive emoticons
plot(tweetRefHasEmoticons[tweetRefHasEmoticons>200,1]/
totalRef[which(tweetRefHasEmoticons>200),])
```

```
#part 1
sort(totalRef[which(tweetRefHasEmoticons>200),], decreasing = TRUE)
sort(results[which(results[,3] > 200),2], decreasing = TRUE) #equivalent

#pos for part 2
 sort(which(tweetRefHasEmoticons>200), decreasing = TRUE)

plot(tweetRefHasEmoticons[ sort(which(tweetRefHasEmoticons>200), decreasing = TRUE),1]/
sort(totalRef[which(tweetRefHasEmoticons>200),], decreasing = TRUE),  xlab = "x", ylab="y")

# Sort the data frame by the third column
var = results[which(results[,3]>30),]                    # filter on column 3
order_var2=var[order(var[,2], decreasing=TRUE),]  # sort on column 2
x_plot = order_var2[,3]/order_var2[,2]

plot(x_plot, main="Relative frequency that @mentioned users coincide with emoticons" ,
xlab="Ranked users by most emoticons" , ylab="Relative frequency of being only @mentioned
vs being @mentioned & emoticon" ,  type="h" ,  lwd=1)
x_length = length(order_var2[,1])
x = 1:x_length
y = order_var2[,3]/order_var2[,2]
lo = loess(y~x)
lines(predict(lo), col='red', lwd=2)
# lines(density(1-y), col='red', lwd=2)
# Interesting graph
# apparently there is something going on with the results..... I used to have
# order_var2[,3]/(order_var2[,2]+order_var2[,3]) but then it goes to 0.5 only...

# Testing on normality
# .1*length(ratio)
var = results[which(results[,3]>30),]
ratio = var[,4]/(var[,4]+var[,5])
histogram=hist(ratio, breaks=40, freq=FALSE)
x = (1:length(histogram$counts))/length(histogram$counts)
y = histogram$counts
lo = loess(y~x)


# Read in smileys
emoticons = read.table(file="/home/melvin/Desktop/IMM/b-these/smileys_count.tsv",
col.names=c("count", "emoticons"), header = TRUE, sep="\t")
plot(emoticons[,1], log="xy", xlab="Emoticons in rankorder of frequency", ylab="Frequency",
main="Frequencies of emoticons in log log scale")

# Read in emoticons in java program
emoticons = read.table(file="/home/melvin/Desktop/IMM/b-these/emoticons.tsv",
col.names=c("count", "emoticons"), header = TRUE, sep="\t")
plot(emoticons[,1], log="xy", xlab="Emoticons in rankorder of frequency", ylab="Frequency",
main="Frequencies of emoticons in log log scale")
```

```
# Read classified emoticons
emoticons_positive = read.table(file="/home/melvin/Desktop/IMM/b-these/
emoticons_positive.tsv", col.names=c("count", "emoticons"), header = TRUE, sep="\t")
emoticons_negative = read.table(file="/home/melvin/Desktop/IMM/b-these/
emoticons_negative.tsv", col.names=c("count", "emoticons"), header = TRUE, sep="\t")
sum(emoticons_positive[,1])/(sum(emoticons_positive[,1])+sum(emoticons_negative[,1]))

# [1] 0.8648454
# > sum(emoticons_positive[,1])
# [1] 17814276
# > sum(emoticons_negative[,1])
# [1] 2783944
```