**2. a.**

- Berapa jumlah variable dari data tersebut?

**Jawab:**

Data tersebut memiliki dua variabel, variabel pertama adalah tweet, variabel kedua adalah class

- Tuliskan code untuk mengetahui jumlah tweet dari data tersebut!  
Jawab:

nrow(tweet\_raw)



- Ubahlah tipe data dari kolom class menjadi tipe data factor

Jawab:

tweet\_raw$class = as.factor(tweet\_raw$class)

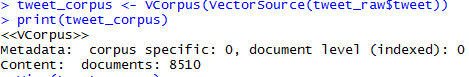


**2. b.**

- Informasi apa yang didapatkan dari perintah print(tweet\_corpus)?

Jawab:

Diperoleh informasi banyaknya tweet pada corpus yaitu 8510 tweet



**2. c.**

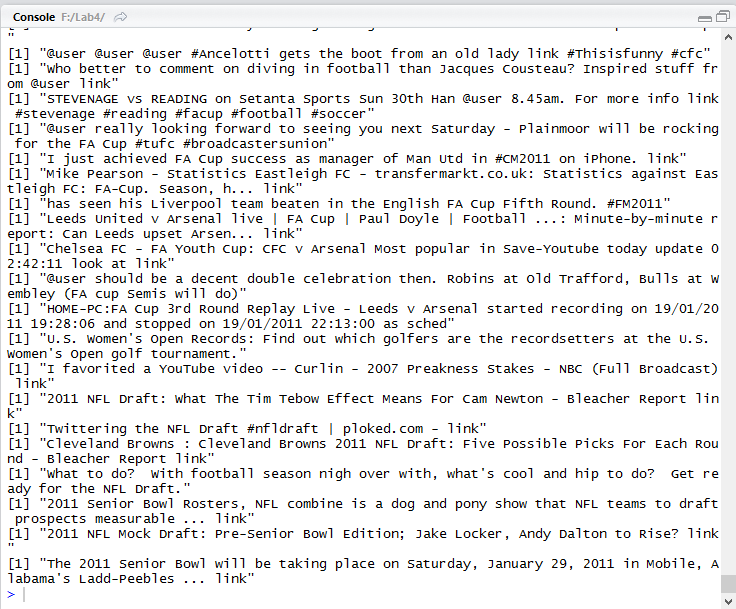
- Tuliskan code untuk melihat isi corpus 50% baris pertama

halflength = length(tweet\_corpus)/2

for (i in 1:halflength){

print(as.character(tweet\_corpus[[i]]))

}



2. d.

- Perlihatkan perbandingan antara sebelum diubah menjadi lowercase dan setelahnya

Jawab:

Sebagai contoh perbandingan, diperlihatkan 10 tweet pertama

tweet\_corpus\_clean = tm\_map(tweet\_corpus,content\_transformer(tolower))

#10 clean tweet pertama

for (i in 1:10){

print(as.character(tweet\_corpus\_clean[[i]]))

}

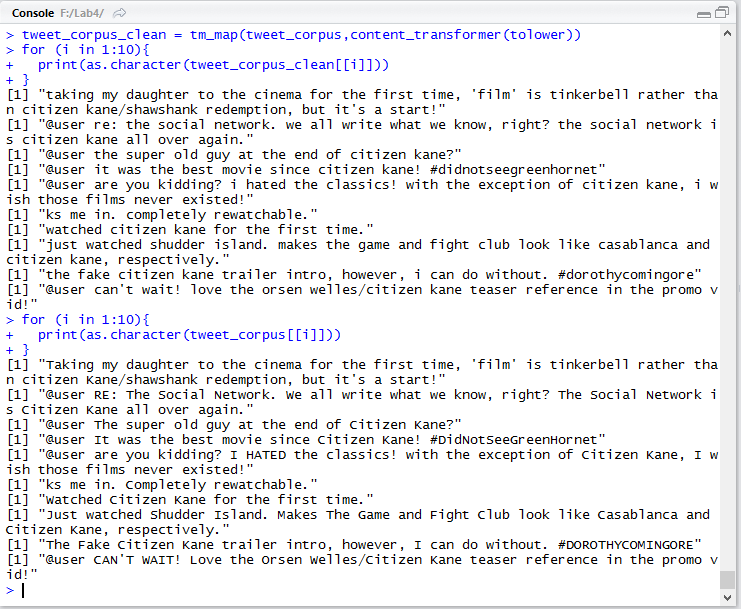
#10 original tweet pertama

for (i in 1:10){

print(as.character(tweet\_corpus[[i]]))

}

Berikut perbandingannya (diperlihatkan contoh perbandingan untuk 10 tweet pertama)



Bagian atas adalah tweet yang sudah dibuat lower case, bagian bawah adalah tweet original.

2. h.

- Sebutkan dan jelaskan 2 kesalahan yang dapat timbul karena proses stemming!

Jawab:

* Kesalahan dalam mendapatkan kata dasar dari kata berimbuhan yang irreguler. Contohnya pada tweet ke 21 pada corpus diatas, kata “better” tidak dapat dilakukan stemming menjadi “good” karena bentuknya irreguler.



* Kesalahan karena terdapat kata tidak berimbuhan yang dianggap berimbuhan sehingga kata tersebut dipotong, padahal seharusnya tidak. Contohnya pada tweet ke 9 pada corpus diatas terdapat kata “however” yang diubah menjadi “howev” saat stemming. Jadi kata “however” dianggap sebagai “howev”+”-er”.



3. b.

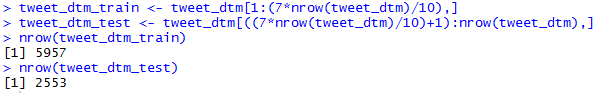
- Bagilah data menjadi 70% data training dan 30% data testing, kemudian simpan labelnya

Jawab:

70% data pertama sebagai training dan 30% data terakhir sebagai testing

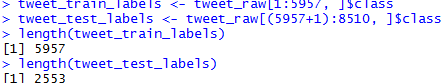
tweet\_dtm\_train <- tweet\_dtm[1:(7\*nrow(tweet\_dtm)/10),]

tweet\_dtm\_test <- tweet\_dtm[((7\*nrow(tweet\_dtm)/10)+1):nrow(tweet\_dtm),]



tweet\_train\_labels <- tweet\_raw[1:5957, ]$class

tweet\_test\_labels <- tweet\_raw[(5957+1):8510, ]$class



**3. c.**

- Buat 2 wordcloud masing-masing untuk kata-kata yang ada dalam dokumen berlabel objnspam dan berlabel objspam. Hisnt: gunakan fungsi subset.

Jawab:

* Wordcloud untuk objnspam

objnspam\_index = c()

for(i in 1:nrow(tweet\_raw)){

if(tweet\_raw[i,]$class == 'objnspam'){

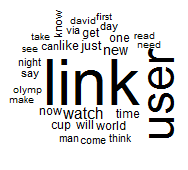
objnspam\_index = c(objnspam\_index, i)

}

}

tweet\_corpus\_clean\_objnspam = tweet\_corpus\_clean[objnspam\_index]

wordcloud(tweet\_corpus\_clean\_objnspam, min.freq = 50, random.order = FALSE)



* Wordcloud untuk objspam

objspam\_index = c()

for(i in 1:nrow(tweet\_raw)){

if(tweet\_raw[i,]$class == 'objspam'){

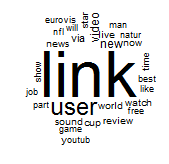
objspam\_index = c(objspam\_index, i)

}

}

tweet\_corpus\_clean\_objspam = tweet\_corpus\_clean[objspam\_index]

wordcloud(tweet\_corpus\_clean\_objspam, min.freq = 30, random.order = FALSE)



**4.**

- Buat fitur boolean (0 atau 1) untuk menandakan adanya kata tersebut pada suatu dokumen. Hint : gunakan fungsi apply

Jawab:

tweet\_train\_boolean <- apply(tweet\_dtm\_freq\_train, 1:2, function(x) if(x>0){1} else{0})

tweet\_test\_boolean <- apply(tweet\_dtm\_freq\_test, 1:2, function(x) if(x>0){1} else{0})

5.

- Sebutkan 5 fitur (dengan pembobotan frekuensi) yang tertinggi bobotnya!

Jawab: link, love, user, like, hate

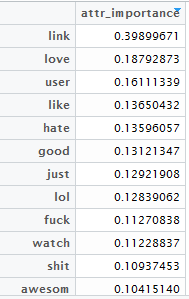
training\_freq = data.frame(as.matrix(tweet\_dtm\_freq\_train))

View(training\_freq)

training\_freq$tweetlabel = tweet\_train\_labels

weights <- chi.squared(tweetlabel ~., training\_freq)

View(weights)



- Sebutkan 5 fitur (dengan pembobotan boolean) yang tertinggi bobotnya. Apakah sama dengan pembobotan frekuensi?Jawab: link, love, user, like, hate

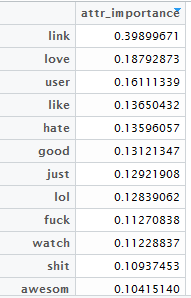
training\_boolean = data.frame(as.matrix(tweet\_train\_boolean))

View(training\_boolean)

training\_boolean$tweetlabel = tweet\_train\_labels

weights\_boolean <- chi.squared(tweetlabel ~., training\_boolean)

View(weights\_boolean)



Kelima fitur tersebut sama dengan saat menggunakan pembobotan frekuensi

- Aplikasikan 1 metode univariat lain yang Anda sudah pelajari!

- Aplikasikan 1 metode multivariat lain yang Anda sudah pelajari!

- Aplikasikan salah satu metode klasifikasi sederhana untuk membandingkan akurasi fiturfitur terseleksi dari kedua metode tersebut! Metode mana yang lebih baik? Berikan analisis Anda!

- Ganti tokenisasi data dengan menggunakan tf-idf, bandingkan dengan menggunakan metode klasifikasi sederhana.

**CITRA**

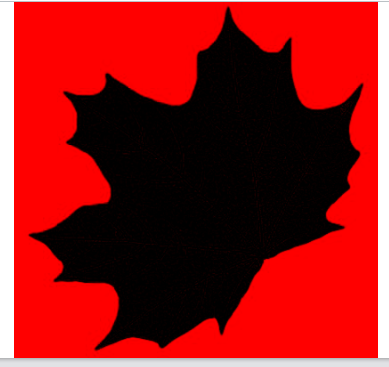
**3. a.**

- Tampilkan citra dengan mengganti nilai parameter ke-3 dengan nilai 3! Properti warna apa yang muncul? Bagaimana dengan nilai 2?

Jawab:

Saat nilai parameter ketiga adalah 3

display(img[,,3], method="raster");



Warna yang muncul hanya dua, hitam dan merah

Saat nilai parameter ketiga adalah 2

display(img[,,2], method="raster");



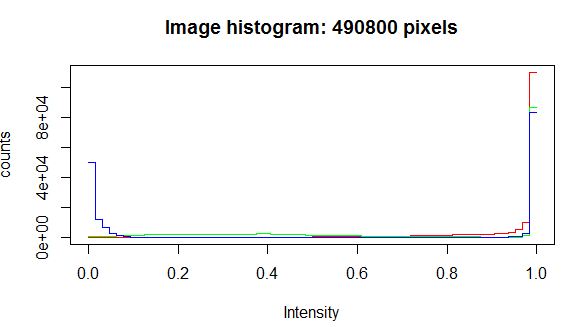
Warna yang muncul adalah merah dan hitam namun lebih banyak variasinya (tidak hanya dua warna)

- Kenapa pada judul histogram tertulis "490800 pixels"?  
Jawab:

Penghitungan pixel tersebut merupakan penghitungan tiga layer RGB sehingga jumlah pixelnya menjadi

- Apa yang dapat Anda analisis dari histogram tersebut (berdasarkan intensitas dan jumlah masing-masing properti warna)?

Jawab:



Dapat dilihat bahwa warna merah memiliki jumlah pixel yang paling banyak diantara warna yang lain serta memiliki intensitas yang paling tinggi. Ini dapat dilihat juga dari gambar daun yang warna merahnya cenderung dominan. Warna lain yang cukup tinggi kuantitasnya adalah biru. Sedangkan warna hijau hampir tidak ada pada gambar tersebut.

- Apa beda struktur data dari "img\_ch\_r" dengan "channels$red" (bukan "channels" melainkan "channels$red")? Kenapa ada perbedaan itu (pahami kode program di atas)?

Jawab:

channels$red disimpan dalam struktur data vector of number satu dimensi, sedangkan img\_ch\_r memiliki struktur image dan datanya disimpan dalam array 2 dimensi dengan ukuran 409x400.

Hal ini terjadi karena pada program tersebut, channels diperoleh dengan menerapkan ‘sapply’ dengan menggunakan ‘as.vector(channel(img, ch)’ sehingga data channels$red berbentuk vektor. Sementara itu img\_ch\_r diperoleh dengan channel(img, "asred") sehingga bentuk datanya tetap array dua dimensi sesuai ukuran resolusi imagenya.

- Hitung mean dari properti RGB citra untuk menjadi nilai fitur?

Jawab:

img\_R\_mean = mean(channels$red)

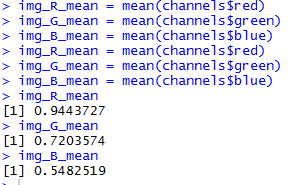
img\_G\_mean = mean(channels$green)

img\_B\_mean = mean(channels$blue)

img\_R\_mean

img\_G\_mean

img\_B\_mean



- Hitung rata-rata dan standar deviasi dari properti HSV citra untuk menjadi nilai fitur seperti dari properti RGB?

Jawab:

img\_H\_sd = sd(img\_hsv[1,])

img\_S\_sd = sd(img\_hsv[2,])

img\_V\_sd = sd(img\_hsv[3,])

img\_H\_sd

img\_S\_sd

img\_V\_sd

img\_H\_mean = mean(img\_hsv[1,])

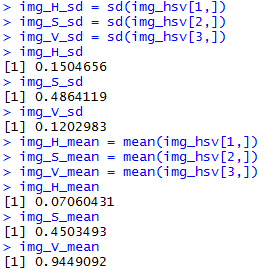
img\_S\_mean = mean(img\_hsv[2,])

img\_V\_mean = mean(img\_hsv[3,])

img\_H\_mean

img\_S\_mean

img\_V\_mean



- Buktikan apakah nilai V sama dengan Cmax.

Jawab:

Keduanya SAMA

cmax\_all\_pixels = pmax(channels$red, channels$green, channels$blue)

identical(cmax\_all\_pixels, img\_hsv[3,])



**- Buktikan apakah nilai S sama dengan delta/Cmax**

**Jawab:**

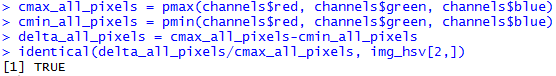
Keduanya SAMA

cmax\_all\_pixels = pmax(channels$red, channels$green, channels$blue)

cmin\_all\_pixels = pmin(channels$red, channels$green, channels$blue)

delta\_all\_pixels = cmax\_all\_pixels-cmin\_all\_pixels

identical(delta\_all\_pixels/cmax\_all\_pixels, img\_hsv[2,])



- Lanjutkan kode program untuk menghitung nilai Y dan Q!

Jawab:

img\_Y <- 0.299 \* channels$red + 0.587 \* channels$green + 0.114\*channels$blue

img\_Q <- 0.211 \* channels$red - 0.523 \* channels$green + 0.312\*channels$blue

- Lanjutkan kode program untuk menghitung rata-rata dan standar deviasi untuk nilai YIQ!

Jawab:

img\_Y\_mean = mean(img\_Y)

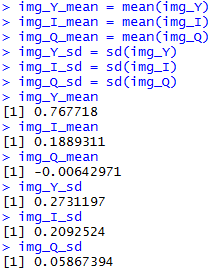
img\_I\_mean = mean(img\_I)

img\_Q\_mean = mean(img\_Q)

img\_Y\_sd = sd(img\_Y)

img\_I\_sd = sd(img\_I)

img\_Q\_sd = sd(img\_Q)



**3. b.**

- Sebagai bahan perbandingan, lakukan langkah yang sama untuk membuat fitur bentuk pada citra daun\_2 dan daun\_3, manakah yang lebih besar nilai area dan perimeternya? Mengapa demikian?

Jawab:

img\_gray = channel(img,"gray")#convert image to grayscale

fts = computeFeatures.shape(img\_gray)#generate shape features

fts

img2 <- readImage("daun\_2.jpg")

img3 <- readImage("daun\_3.jpg")

img\_gray2 = channel(img2,"gray")#convert image to grayscale

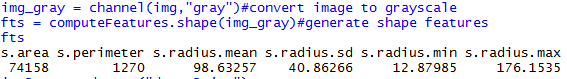
fts2 = computeFeatures.shape(img\_gray2)#generate shape features

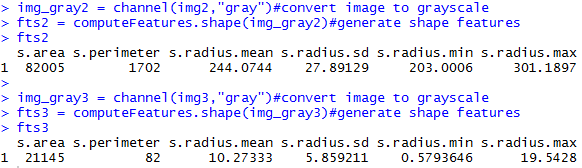
fts2

img\_gray3 = channel(img3,"gray")#convert image to grayscale

fts3 = computeFeatures.shape(img\_gray3)#generate shape features

fts3





Dari hasil tersebut dapat diketahui bahwa area dan perimeter terbesar keduanya dimiliki oleh gambar daun\_2. Hal ini dapat dikarenakan resolusi citra daun\_2 merupakan yang tertinggi dari tiga citra tersebut, artinya jumlah pixelnya lebih banyak sehingga dapat meningkatkan area dan permeter.

- Lakukan plotting untuk nilai tekstur yang lain seperti homogeneity, contrast, dissimiliarity, entropy, second\_moment, correlation.

**Jawab:**

plot(textures$glcm\_homogeneity)

plot(textures$glcm\_contrast)

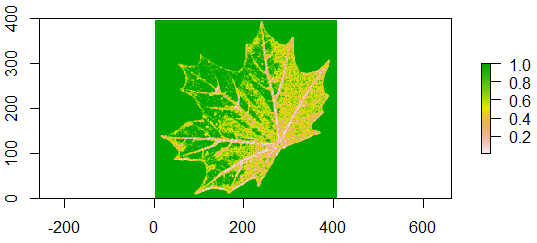
plot(textures$glcm\_dissimilarity)

plot(textures$glcm\_entropy)

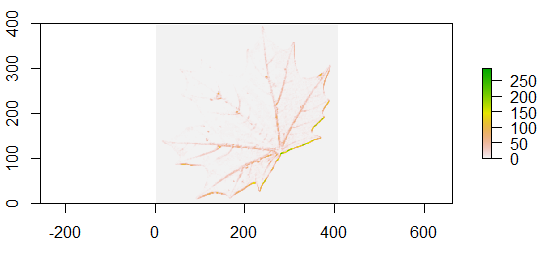
plot(textures$glcm\_second\_moment)

plot(textures$glcm\_correlation)

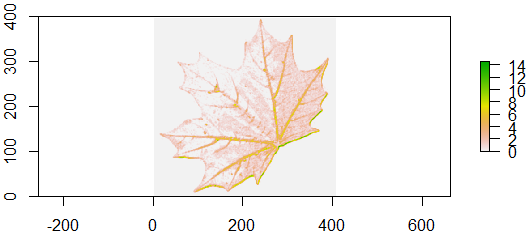
* **Homogenity**



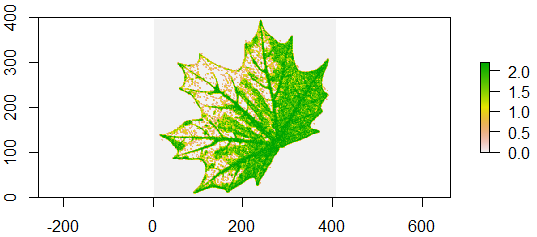
* **Contrast**



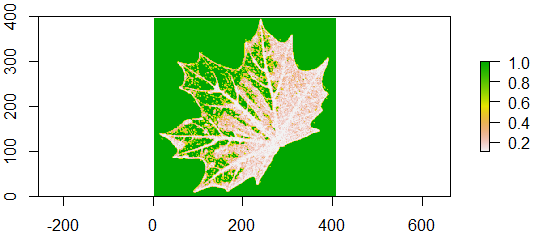
* **Dissimilarity**



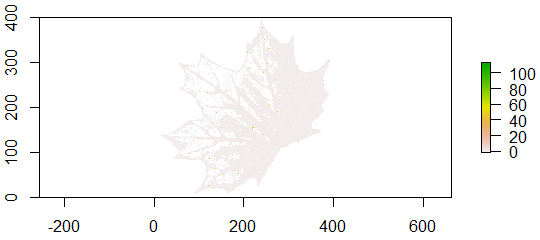
* **Entropy**



* **Second\_moment**



* **Correlation**



**4.**

**Membuat dataframe df**

img1 <- readImage("daun\_1.jpg")

img2 <- readImage("daun\_2.jpg")

img3 <- readImage("daun\_3.jpg")

channels1 = sapply(c("red", "green", "blue"),

function(ch) as.vector(channel(img1, ch)),

simplify = FALSE)

channels1 = as.data.frame(channels1)

channels2 = sapply(c("red", "green", "blue"),

function(ch) as.vector(channel(img2, ch)),

simplify = FALSE)

channels2 = as.data.frame(channels2)

channels3 = sapply(c("red", "green", "blue"),

function(ch) as.vector(channel(img3, ch)),

simplify = FALSE)

channels3 = as.data.frame(channels3)

img1\_R\_mean = mean(channels1$red)

img1\_G\_mean = mean(channels1$green)

img1\_B\_mean = mean(channels1$blue)

img2\_R\_mean = mean(channels2$red)

img2\_G\_mean = mean(channels2$green)

img2\_B\_mean = mean(channels2$blue)

img3\_R\_mean = mean(channels3$red)

img3\_G\_mean = mean(channels3$green)

img3\_B\_mean = mean(channels3$blue)

img1\_hsv <- rgb2hsv(channels1$red\*255,channels1$green\*255,

channels1$blue\*255, maxColorValue = 255);

img2\_hsv <- rgb2hsv(channels2$red\*255,channels2$green\*255,

channels2$blue\*255, maxColorValue = 255);

img3\_hsv <- rgb2hsv(channels3$red\*255,channels3$green\*255,

channels3$blue\*255, maxColorValue = 255);

img1\_H\_mean = mean(img1\_hsv[1,])

img1\_S\_mean = mean(img1\_hsv[2,])

img1\_V\_mean = mean(img1\_hsv[3,])

img2\_H\_mean = mean(img2\_hsv[1,])

img2\_S\_mean = mean(img2\_hsv[2,])

img2\_V\_mean = mean(img2\_hsv[3,])

img3\_H\_mean = mean(img3\_hsv[1,])

img3\_S\_mean = mean(img3\_hsv[2,])

img3\_V\_mean = mean(img3\_hsv[3,])

img1\_Y <- 0.299 \* channels1$red + 0.587 \* channels1$green + 0.114\*channels1$blue

img1\_I <- 0.596 \* channels1$red - 0.274 \* channels1$green - 0.322\*channels1$blue

img1\_Q <- 0.211 \* channels1$red - 0.523 \* channels1$green + 0.312\*channels1$blue

img2\_Y <- 0.299 \* channels2$red + 0.587 \* channels2$green + 0.114\*channels2$blue

img2\_I <- 0.596 \* channels2$red - 0.274 \* channels2$green - 0.322\*channels2$blue

img2\_Q <- 0.211 \* channels2$red - 0.523 \* channels2$green + 0.312\*channels2$blue

img3\_Y <- 0.299 \* channels3$red + 0.587 \* channels3$green + 0.114\*channels3$blue

img3\_I <- 0.596 \* channels3$red - 0.274 \* channels3$green - 0.322\*channels3$blue

img3\_Q <- 0.211 \* channels3$red - 0.523 \* channels3$green + 0.312\*channels3$blue

img1\_Y\_mean = mean(img1\_Y)

img1\_I\_mean = mean(img1\_I)

img1\_Q\_mean = mean(img1\_Q)

img2\_Y\_mean = mean(img2\_Y)

img2\_I\_mean = mean(img2\_I)

img2\_Q\_mean = mean(img2\_Q)

img3\_Y\_mean = mean(img3\_Y)

img3\_I\_mean = mean(img3\_I)

img3\_Q\_mean = mean(img3\_Q)

img1\_gray = channel(img1,"gray")#convert image to grayscale

fts1 = computeFeatures.shape(img1\_gray)#generate shape features

img2\_gray = channel(img2,"gray")#convert image to grayscale

fts2 = computeFeatures.shape(img2\_gray)#generate shape features

img3\_gray = channel(img3,"gray")#convert image to grayscale

fts3 = computeFeatures.shape(img3\_gray)#generate shape features

img1\_area = fts1[1]

img1\_perimeter = fts1[2]

img1\_radius\_mean = fts1[3]

img1\_radius\_sd = fts1[4]

img1\_radius\_min = fts1[5]

img1\_radius\_max = fts1[6]

img2\_area = fts2[1]

img2\_perimeter = fts2[2]

img2\_radius\_mean = fts2[3]

img2\_radius\_sd = fts2[4]

img2\_radius\_min = fts2[5]

img2\_radius\_max = fts2[6]

img3\_area = fts3[1]

img3\_perimeter = fts3[2]

img3\_radius\_mean = fts3[3]

img3\_radius\_sd = fts3[4]

img3\_radius\_min = fts3[5]

img3\_radius\_max = fts3[6]

r1 <- raster("daun\_1.jpg") # need "rgdal" package

textures1 <- glcm(r1, window = c(3, 3), shift = c(1, 1), statistics =

c("mean", "mean\_ENVI", "variance", "variance\_ENVI", "homogeneity",

"contrast", "dissimilarity", "entropy", "second\_moment",

"correlation"), na\_opt="any", na\_val=NA)

r2 <- raster("daun\_2.jpg") # need "rgdal" package

textures2 <- glcm(r2, window = c(3, 3), shift = c(1, 1), statistics =

c("mean", "mean\_ENVI", "variance", "variance\_ENVI", "homogeneity",

"contrast", "dissimilarity", "entropy", "second\_moment",

"correlation"), na\_opt="any", na\_val=NA)

r3 <- raster("daun\_3.jpg") # need "rgdal" package

textures3 <- glcm(r3, window = c(3, 3), shift = c(1, 1), statistics =

c("mean", "mean\_ENVI", "variance", "variance\_ENVI", "homogeneity",

"contrast", "dissimilarity", "entropy", "second\_moment",

"correlation"), na\_opt="any", na\_val=NA)

img1\_glcm\_mean = (minValue(textures1$glcm\_mean)+maxValue(textures1$glcm\_mean))/2

img1\_glcm\_variance = (minValue(textures1$glcm\_variance)+maxValue(textures1$glcm\_variance))/2

img1\_glcm\_homogenity = (minValue(textures1$glcm\_homogeneity)+maxValue(textures1$glcm\_homogeneity))/2

img1\_glcm\_contrast = (minValue(textures1$glcm\_contrast)+maxValue(textures1$glcm\_contrast))/2

img1\_glcm\_dissimilarity = (minValue(textures1$glcm\_dissimilarity)+maxValue(textures1$glcm\_dissimilarity))/2

img1\_glcm\_entropy = (minValue(textures1$glcm\_entropy)+maxValue(textures1$glcm\_entropy))/2

img1\_glcm\_second\_moment = (minValue(textures1$glcm\_second\_moment)+maxValue(textures1$glcm\_second\_moment))/2

img1\_glcm\_correlation = (minValue(textures1$glcm\_correlation)+maxValue(textures1$glcm\_correlation))/2

img2\_glcm\_mean = (minValue(textures2$glcm\_mean)+maxValue(textures2$glcm\_mean))/2

img2\_glcm\_variance = (minValue(textures2$glcm\_variance)+maxValue(textures2$glcm\_variance))/2

img2\_glcm\_homogenity = (minValue(textures2$glcm\_homogeneity)+maxValue(textures2$glcm\_homogeneity))/2

img2\_glcm\_contrast = (minValue(textures2$glcm\_contrast)+maxValue(textures2$glcm\_contrast))/2

img2\_glcm\_dissimilarity = (minValue(textures2$glcm\_dissimilarity)+maxValue(textures2$glcm\_dissimilarity))/2

img2\_glcm\_entropy = (minValue(textures2$glcm\_entropy)+maxValue(textures2$glcm\_entropy))/2

img2\_glcm\_second\_moment = (minValue(textures2$glcm\_second\_moment)+maxValue(textures2$glcm\_second\_moment))/2

img2\_glcm\_correlation = (minValue(textures2$glcm\_correlation)+maxValue(textures2$glcm\_correlation))/2

img3\_glcm\_mean = (minValue(textures3$glcm\_mean)+maxValue(textures3$glcm\_mean))/2

img3\_glcm\_variance = (minValue(textures3$glcm\_variance)+maxValue(textures3$glcm\_variance))/2

img3\_glcm\_homogenity = (minValue(textures3$glcm\_homogeneity)+maxValue(textures3$glcm\_homogeneity))/2

img3\_glcm\_contrast = (minValue(textures3$glcm\_contrast)+maxValue(textures3$glcm\_contrast))/2

img3\_glcm\_dissimilarity = (minValue(textures3$glcm\_dissimilarity)+maxValue(textures3$glcm\_dissimilarity))/2

img3\_glcm\_entropy = (minValue(textures3$glcm\_entropy)+maxValue(textures3$glcm\_entropy))/2

img3\_glcm\_second\_moment = (minValue(textures3$glcm\_second\_moment)+maxValue(textures3$glcm\_second\_moment))/2

img3\_glcm\_correlation = (minValue(textures3$glcm\_correlation)+maxValue(textures3$glcm\_correlation))/2

###Dibuat 0 karena nilainya NaN

img1\_glcm\_correlation = 0

img2\_glcm\_correlation = 0

img3\_glcm\_correlation = 0

R\_mean = c(img1\_R\_mean, img2\_R\_mean, img3\_R\_mean)

G\_mean = c(img1\_G\_mean, img2\_G\_mean, img3\_G\_mean)

B\_mean = c(img1\_B\_mean, img2\_B\_mean, img3\_B\_mean)

H\_mean = c(img1\_H\_mean, img2\_H\_mean, img3\_H\_mean)

S\_mean = c(img1\_S\_mean, img2\_S\_mean, img3\_S\_mean)

V\_mean = c(img1\_V\_mean, img2\_V\_mean, img3\_V\_mean)

Y\_mean = c(img1\_Y\_mean, img2\_Y\_mean, img3\_Y\_mean)

I\_mean = c(img1\_I\_mean, img2\_I\_mean, img3\_I\_mean)

Q\_mean = c(img1\_Q\_mean, img2\_Q\_mean, img3\_Q\_mean)

area = c(img1\_area, img2\_area, img3\_area)

perimeter = c(img1\_perimeter, img2\_perimeter, img3\_perimeter)

radius\_mean = c(img1\_radius\_mean, img2\_radius\_mean, img3\_radius\_mean)

radius\_sd = c(img1\_radius\_sd, img2\_radius\_sd, img3\_radius\_sd)

radius\_min = c(img1\_radius\_min, img2\_radius\_min, img3\_radius\_min)

radius\_max = c(img1\_radius\_max, img2\_radius\_max, img3\_radius\_max)

glcm\_mean = c(img1\_glcm\_mean, img2\_glcm\_mean, img3\_glcm\_mean)

glcm\_variance = c(img1\_glcm\_variance, img2\_glcm\_variance, img3\_glcm\_variance)

glcm\_homogenity = c(img1\_glcm\_homogenity, img2\_glcm\_homogenity, img3\_glcm\_homogenity)

glcm\_contrast = c(img1\_glcm\_contrast, img2\_glcm\_contrast, img3\_glcm\_contrast)

glcm\_dissimilarity = c(img1\_glcm\_dissimilarity, img2\_glcm\_dissimilarity, img3\_glcm\_dissimilarity)

glcm\_entropy = c(img1\_glcm\_entropy, img2\_glcm\_entropy, img3\_glcm\_entropy)

glcm\_second\_moment = c(img1\_glcm\_second\_moment, img2\_glcm\_second\_moment, img3\_glcm\_second\_moment)

glcm\_correlation = c(img1\_glcm\_correlation, img2\_glcm\_correlation, img3\_glcm\_correlation)

Kelas = as.factor(c('A','B','C'))

df = data.frame(Kelas, R\_mean, G\_mean, B\_mean, H\_mean, S\_mean, V\_mean, Y\_mean,

I\_mean, Q\_mean, area, perimeter, radius\_mean, radius\_sd,

radius\_min, radius\_max, glcm\_mean, glcm\_variance,

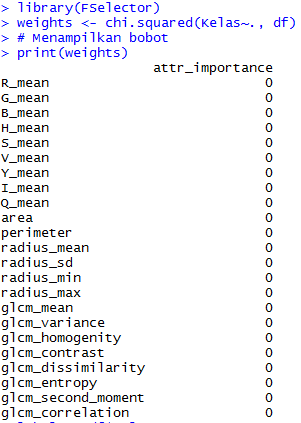
glcm\_homogenity, glcm\_contrast, glcm\_dissimilarity,

glcm\_entropy, glcm\_second\_moment, glcm\_correlation)

- Sebutkan 5 fitur yang memiliki bobot tertinggi!

Jawab:

Semua fitur memiliki bobot 0 karena datanya numerik sehingga tidak cocok digunakan chi-squared



- Sebutkan 5 fitur yang memiliki hasil statistik t-test tertinggi!

**Jawab:**

- Aplikasikan 1 metode univariat lain yang Anda sudah pelajari!  
**Jawab:**

- Aplikasikan 1 metode multivariat lain yang Anda sudah pelajari!  
**Jawab:**

- Aplikasikan salah satu metode klasifikasi sederhana untuk membandingkan akurasi fitur-fitur terseleksi dari metode-metode tersebut! Metode mana yang lebih baik? Berikan analisis Anda!

**Jawab:**