HW6 Solutions

Doug nychka

2/27/2017

-

#Problem 1

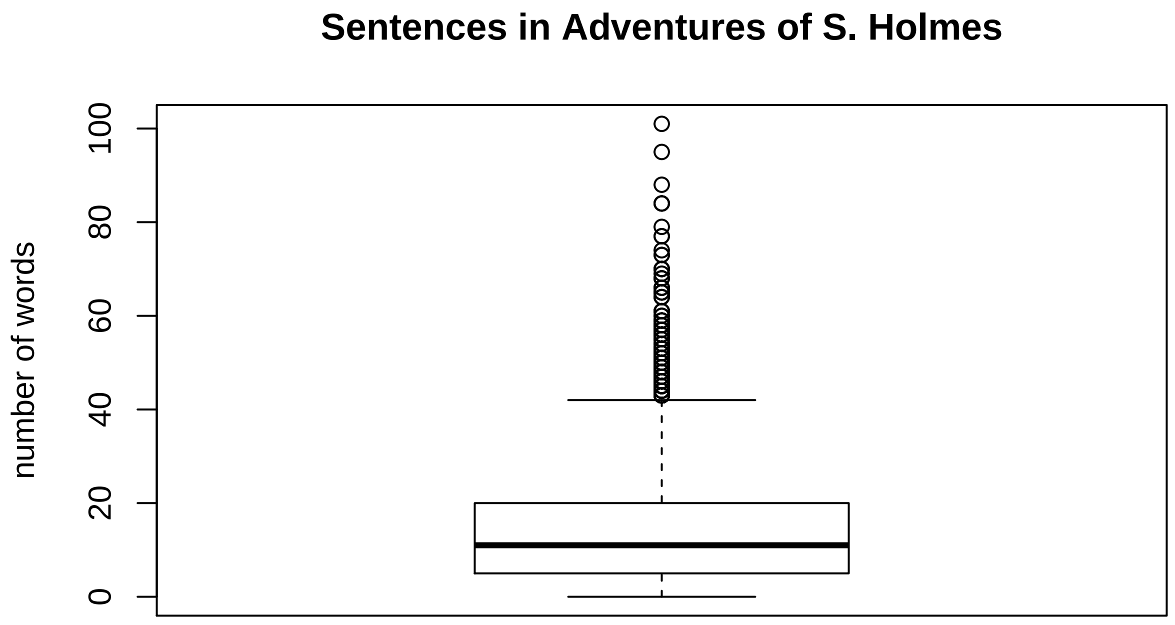
To simply the solutions I added the cleaned text data file `everything4` as the file `everything4.rda` in the class Homework folder.

```
load("everything4.rda)
```

```
#
# number of words (subtract off XXX for periods)
length(everything4) - sum(everything4 =="XXX" )
```

```
## [1] 104897
```

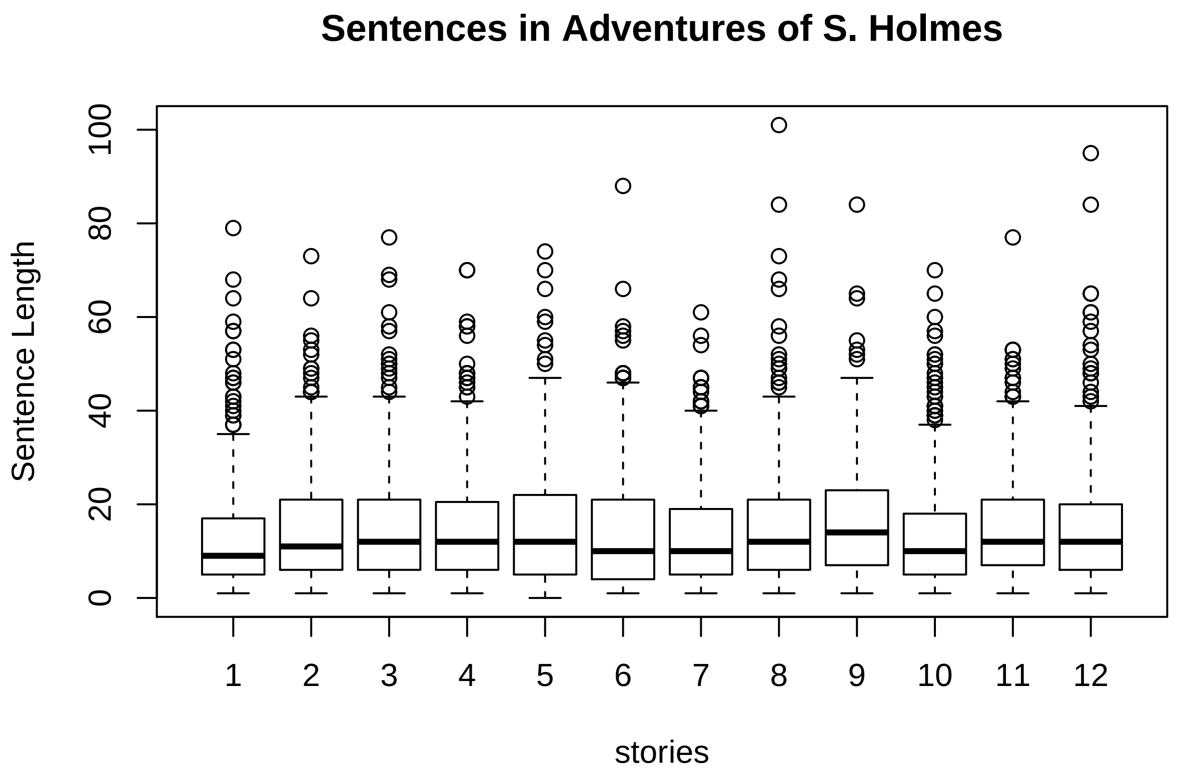
```
 sentenceIndex<- grep("XXX", everything4)
 sentenceLength<- diff(c(1,sentenceIndex) ) - 1
# attach the extra 1 so that the first sentence is counted
# subtract 1 because every sentence has  at least
# XXX -- not a word
boxplot(sentenceLength,ylab="number of words" )
title("Sentences in Adventures of S. Holmes")
```



```
ind<- grep("ADVENTURE", everything4)
ind2<- c( ind, (length(everything4)+1) ) # extra value
# added as a "ghost" beginning of 13th story so that the
# count for the 12th is right.
diffInd<- diff(ind2) # number of words in each story
storyIndex<- rep( 1:12, diffInd)
```

Recall sentences are indexed by `sentenceIndex` This is actually the location of the period at the end of each sentence.

```
#deceptively simple once indices are right!
boxplot(sentenceLength ~ storyIndex[sentenceIndex],
        xlab="stories", ylab="Sentence Length")
title("Sentences in Adventures of S. Holmes")
```



#Problem 2

In working this problem I supressed many of the intermediate image plots so this solution would not get too long.

```
library( jpeg)
CUImage<- readJPEG("CUBoulder.pdf")
# image.plot with helpful coordinates
library( fields)
```

Check out blue color channel.

```
# blue color channel
image.plot(1:376, 1:420, CUImage[,,3] )
```

Try first thresholding the blue channel – this does not work well.

```
#look<- averageImage(CUImage[,,3])
look<- (CUImage[,,3])
look[look >.9  ] <- 1
look[look <=.9 ] <- 0
image.plot(1:376, 1:420,look, col=c( "grey", "blue"))
```

Now try averaging the pixel colors to create less variation.

Here is a simple function to average a pixel block that is 2*Q +1 by 2*Q +1. E.g. Q=2 means a 5x5 block

```r
averageImage<- function(I,Q=2){
  m<- nrow(I)
  n<- ncol(I)
  out<- matrix( 0,nrow=m,ncol=n) # initialize to zero.
  for (  i in (Q+1): (m-Q)){
    for(  j in (Q+1): (n-Q)){
      # average the 5x5 set of pixels centered at i,j
      out[i,j]<-  mean( I[ (i-Q):(i+Q), (j-Q):(j+Q)])
    }
  }
  return( out)
}
```

Average the blue channel image first then find the high spots in the image. The Q and .9 below are tuned to give good results.

```r
# average blue channel  over 5x5 pixels
look<- CUImage
look[,,3]<- averageImage(CUImage[,,3],5 )
look2<- look[,,3]
look2[look2>.9]<- 1
look2[look2 <= .9]<- 0
# check image.plot(1:376, 1:420, look2)
```

Now find the pixel coordinates for the high spots

```r
I<- vector()
J<- vector()
for( i in 1:376){
  for( j in 1:420){
    if( look2[i,j] == 1 ){
# accumulate the I and J coordnates in two vectors
      I<- c( i,I)
      J<- c( j,J)
    }
    }
}
```

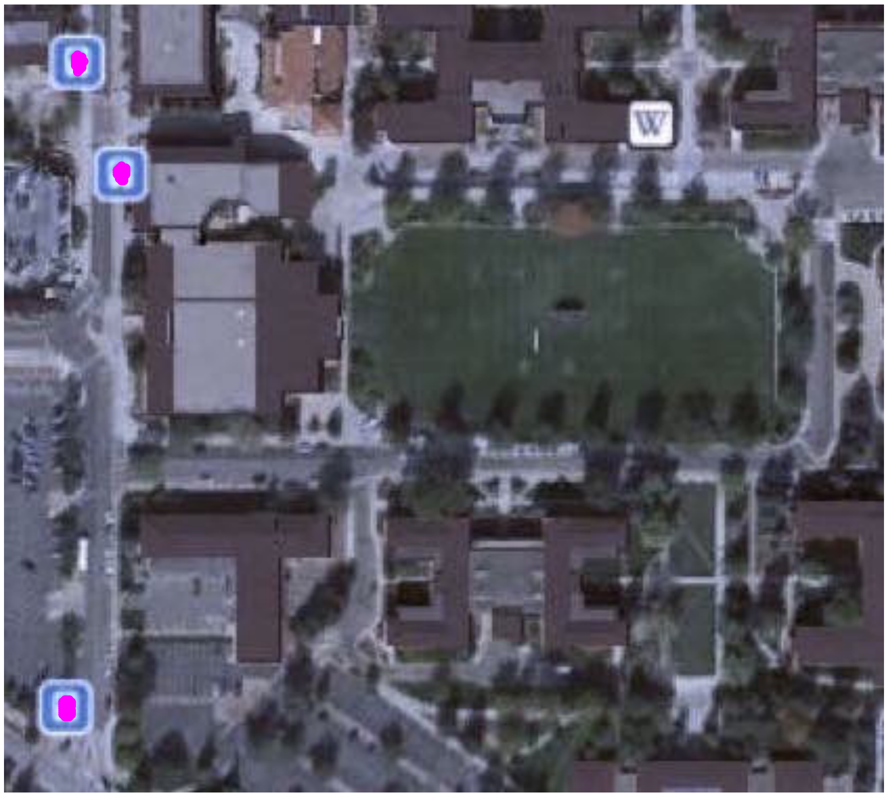
Final image. In image format:

```r
image.plot( 1:376,1:420,CUImage[,,3], col= grey.colors(256) )
points( I,J, col="magenta", cex=.5)
```

Now plotting in the raster format. Note how the pixel locations are switched and flipped.

```r
par( mar=c(0,0,2,0))
plot( as.raster( CUImage))
Iraster<- J # flipping x and y
Jraster<-  376 - I + 1 # reversing the  old x
points(Iraster, Jraster, col="magenta",cex=.5 )
title("Bus stop pixels found (magenta)")
```

Bus stop pixels found (magenta)



Finding roofs in the image. First step was to use the locator to find a rectangle of pixels that were representative of a roof.

```
image.plot( 1:376,1:420,CUImage[,,2], col= grey.colors(256) )
coords<- locator(2)
```

Found

```
ix<- 272:302
iy<- 117:128

# check it
plot( as.raster( CUImage))
newy<- 376 - ix +1
rect( min(iy),min(newy),
      max( iy), max( newy), col="transparent",
      border="magenta")
```

Now find the average color across the color channels for the target. This method might work better by sampling several different roofs that are facing different directions. But to keep this simple I just choose one.

```
meanColor<- c( mean( CUImage[ix,iy,1]),
              mean( CUImage[ix,iy,2]),
              mean( CUImage[ix,iy,3])
              )
tempCol<- rgb(meanColor[1], meanColor[2],meanColor[3])
# plot( 1:4, 1:4, cex=10 , pch=16, col=tempCol)
```

Now find the squared difference between this color and every pixel.

```
look<- (CUImage[,,1] - meanColor[1])^2  +
       (CUImage[,,2] - meanColor[2])^2  +
       (CUImage[,,3] - meanColor[3])^2
```

Check it out

```
image.plot( 1:376,1:420, look )
rect( min(ix),min(iy),
      max( ix), max( iy), col="transparent",
      border="magenta")
```

Now threshold a smoothed image. The tricky way is to calibrate on the same rectangle used to sample colors. Note the tuning parameters here are the Q=2 in the averaging and the 1.5 that inflates the threshold.

```
look2<- look
look2<- averageImage(look,Q = 2 )
threshold<- max( look2[ix,iy])
look3<-  CUImage
# set roofs to pure white
for( j in 1:376) {
 for( k in 1:420){
   if(  look2[j,k]< threshold*1.5 ){
     for( L in 1:3){
   look3[j,k,L]<- 1
     }
 }
 }
}
par( mar=c(0,0,2,0))
plot( as.raster( look3))
newy<- 376 - ix +1
rect( min(iy),min(newy),
      max( iy), max( newy), col="transparent",
      border="magenta")
title("Roofs found by smoothing/threshold algorithm")
```

Roofs found by smoothing/threshold algorithm