Lab 7: Experimenting with Facial Reconstruction using PCA

Introduction to Data Mathematics 2019

Manal Hajjam

Overview

In this lab you'll take a deeper dive into using PCA for image compression and reconstruction. Start the lab by saving the master `.Rmd` file to your local directory and editing the header to include your name.

Preparation: .

Prepare the train and test sets for the face data and do PCA as in PreLab.

As in Prelab 7, we'll use the dataset, `faces.csv`. The following code reads the data into the dataframe `F.df`. The vector `labels` contains numbers representing the identities of the people. The image vectors appear as rows in `F.matrix`. Each image is assigned a row name which is the order the image occurs in the database.

```r
# Read in data
F.df <- read.csv('~/MATP-4400/data/faces.csv')
# Save first column as labels
labels <- as.numeric(F.df[,1])
F.matrix<-as.matrix(F.df[,-1])
# Assign the row names
row.names(F.matrix)<-1:(nrow(F.matrix))
```

You will need the helper functions from Lab 6. The `faceplot()` function draws one or more vectors as an images in a grid.

```r
# A function to help plot faces
faceplot <- function(xx,width=64,midcolor="grey10",gcols=2,labels=row.names(xx)) {
 if (is.vector(xx)) {
   xx <- matrix(xx,nrow=1)
}
  pl <- vector("list",nrow(xx))
  for(i in 1:nrow(xx)){
    face <- matrix(xx[i,], nrow=width)
    face.m <- melt(apply(face, 2, function(x) as.numeric(rev(x))))
    pl[[i]] <- ggplot(data=face.m) + geom_raster(aes(x=Var2, y=Var1, fill=value)) +
        theme(axis.text=element_blank(), axis.title=element_blank()) + guides(fill=FALSE) +
        scale_fill_gradient2(low="black", mid=midcolor, high="white") +ggtitle(labels[i])
  }
  grid.arrange(grobs=pl,ncol=gcols)
}
```

We once again need to generate training and testing datasets. In this lab we are creating a PCA to compress data. Our training data will be used to construct the compression method. We will evaluate how well it works by testing it on the testing data.

As in Lab 6, we'll divide the data into two *disjoint* sets called the training set and the testing set. The data analytics model is created based on the training set. Then the model is applied to the testing set. In the following example, the training set consists of six images for each person and the testing set consists of four images from each person.

```
# Divide into trainface and  testface
# Since images occur in blocks of 10, work mod10 (using %%10) and get  images 4-9 as train and 0-3 as t
gg <- c(1:400)
h <- gg[ gg%%10 >3]
#h is the list of numbers of faces to extract
trainface.uncentered<- F.matrix[h,]
trainfacelabels<-labels[h]
# Test face
gg <- c(1:400)
h <- gg[ gg%%10 <= 3  ]
# h is the list of numbers of faces to extract
testface.uncentered<- F.matrix[h,]
testfacelabels<-labels[h]
```

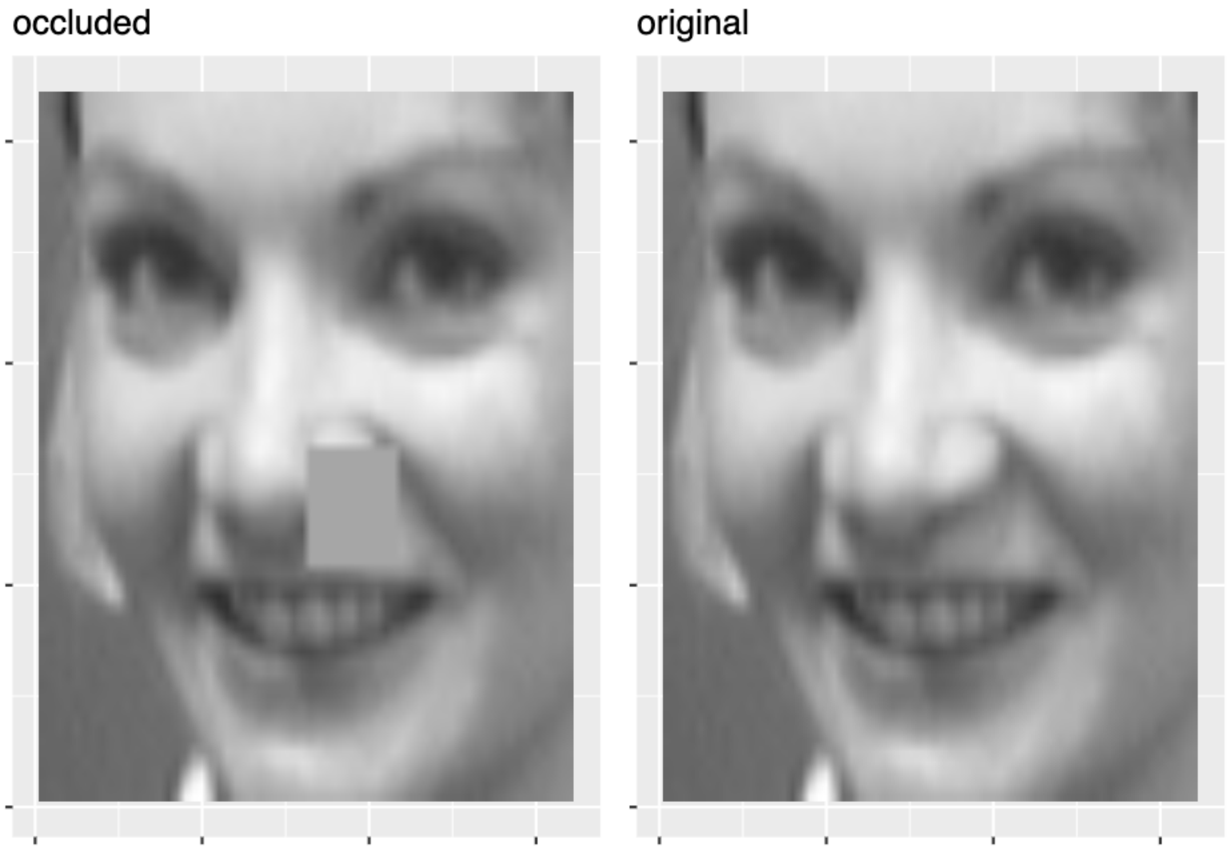
We now perform the PCA analysis we learned in Lab 6 on our training data. First we *mean scale* the data to create the matrix `trainface`. The mean of the training data is saved in `train.mean`.

```
# Find the mean face and plot it
ones<- matrix(1,ncol=1,nrow=nrow(trainface.uncentered))
train.mean <- 1/nrow(trainface.uncentered)* t(ones) %*% trainface.uncentered
# Mean center the data and save in trainface.
trainface<-trainface.uncentered-matrix(1,ncol=1,nrow=nrow(trainface.uncentered))%*%train.mean
# Principal component command. Do not recenter data.
my.pca<- prcomp(trainface,retx=TRUE,center=FALSE)
```

Exercise 1

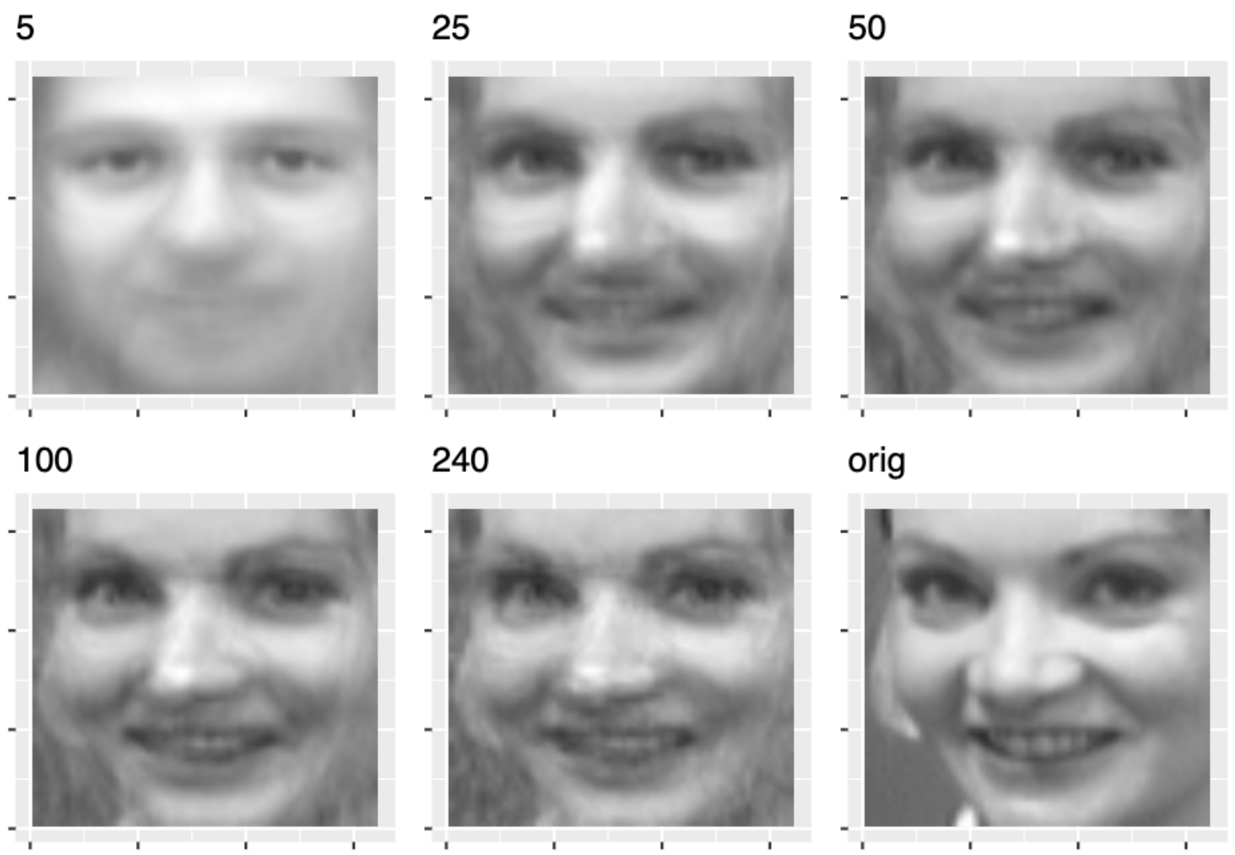
In the following, we construct an image `test.occluded` that is partially blocked by assigning a value of `133` to a block of pixels.

```
imagenum<-139
testimage<-testface.uncentered[imagenum,]
indices<-33:43
testmatrix<-matrix(testimage,nrow=64)
testmatrix[indices,indices]<-133 # This is our occlusion!
test.occluded <-matrix(testmatrix,nrow=1)
pl<-rbind(test.occluded,testface.uncentered[imagenum,])
faceplot(as.data.frame(pl),gcols=2,labels=c('occluded','original'))
```



Reconstruct the original image from **`test.occluded`** using 5, 25,50, 100, and 240 eigenvectors and plot the resulting reconstruction along with the original image. Comment on what happens to the occluded pixels in the reconstructed images. *How well are any occluded parts recovered?*

```
tface<-139
# newface is the centered  test face
newface<-testface.uncentered[tface,]-train.mean
# Get the scalar projections of newface.
scalarproj <- newface %*% my.pca$rotation
recon<- t(my.pca$rotation)  # create the recon matrix as eigenvectors and then replace with reconstruct
# n is the number of eigenvectors found.
n<-nrow(recon)
recon[1,] <- train.mean+scalarproj[1]*recon[1, ]  #Reconstruct uisng the first eigenvector
# Add s_i v_i to each row.
for(j in 2:n){ recon[j, ] <- recon[j-1, ] + scalarproj[j]*recon[j, ]}
# Plot selected reconstructions followed by the original image
#test.occluded <-matrix(testmatrix,nrow=1)
pl <- rbind(recon[c(5,25,50,100,n), ])
pl <- rbind(pl,testface.uncentered[tface,])
# Plot the reconstructed faces for using 5,25,50, 100, the maximum eigenvetors, and the origial image
faceplot(as.data.frame(pl),gcols=3,labels=c('5','25','50','100',as.character(n),'orig'))
```



Exercise 2

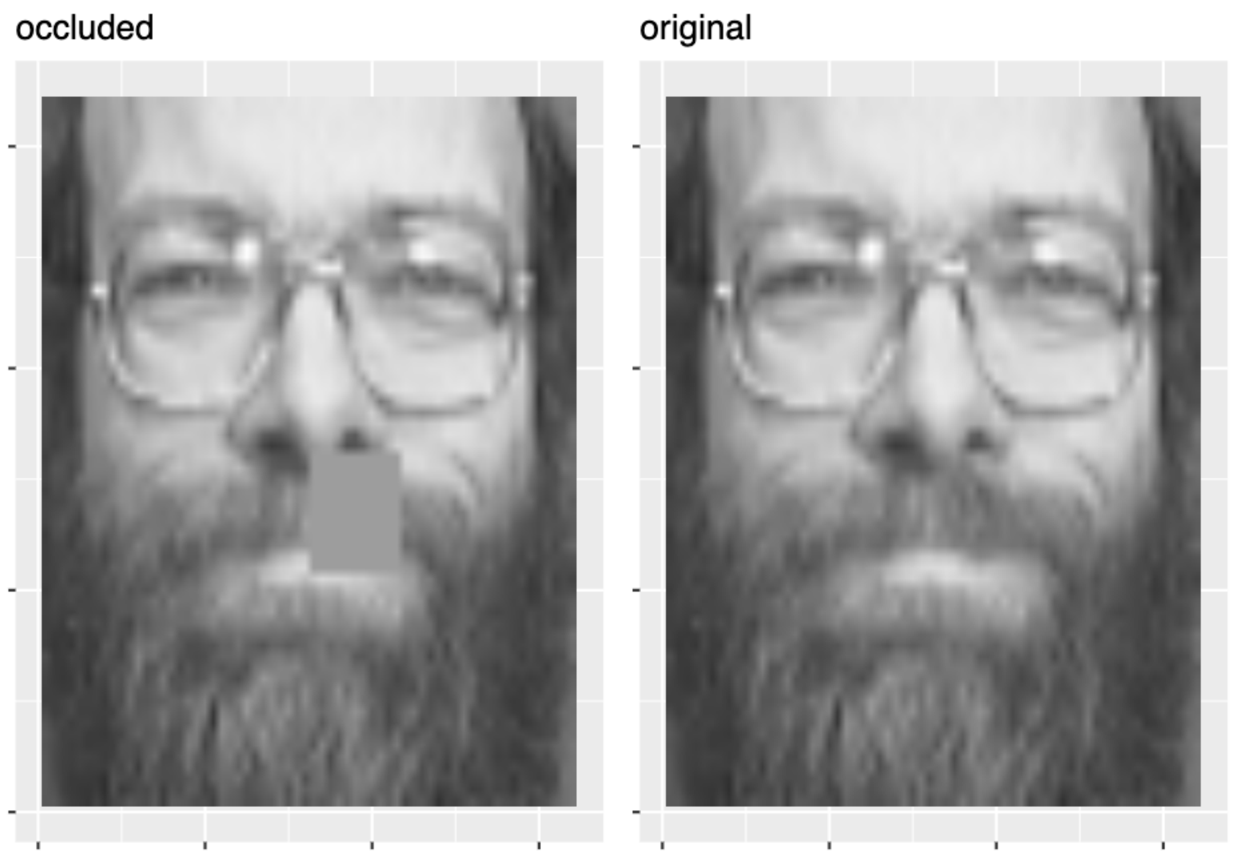
- Pick an image from the testing set.

- Occlude 2048 pixels of your choice by assigning them a value of 133.
- Plot the occluded image and the original image.
- Reconstruct your occluded image using `5`, `25`,`50`, `100`, and `240` eigenvectors and plot the resulting reconstruction along with the original image.
- Comment on what happens to the occlusion on the reconstructed images.

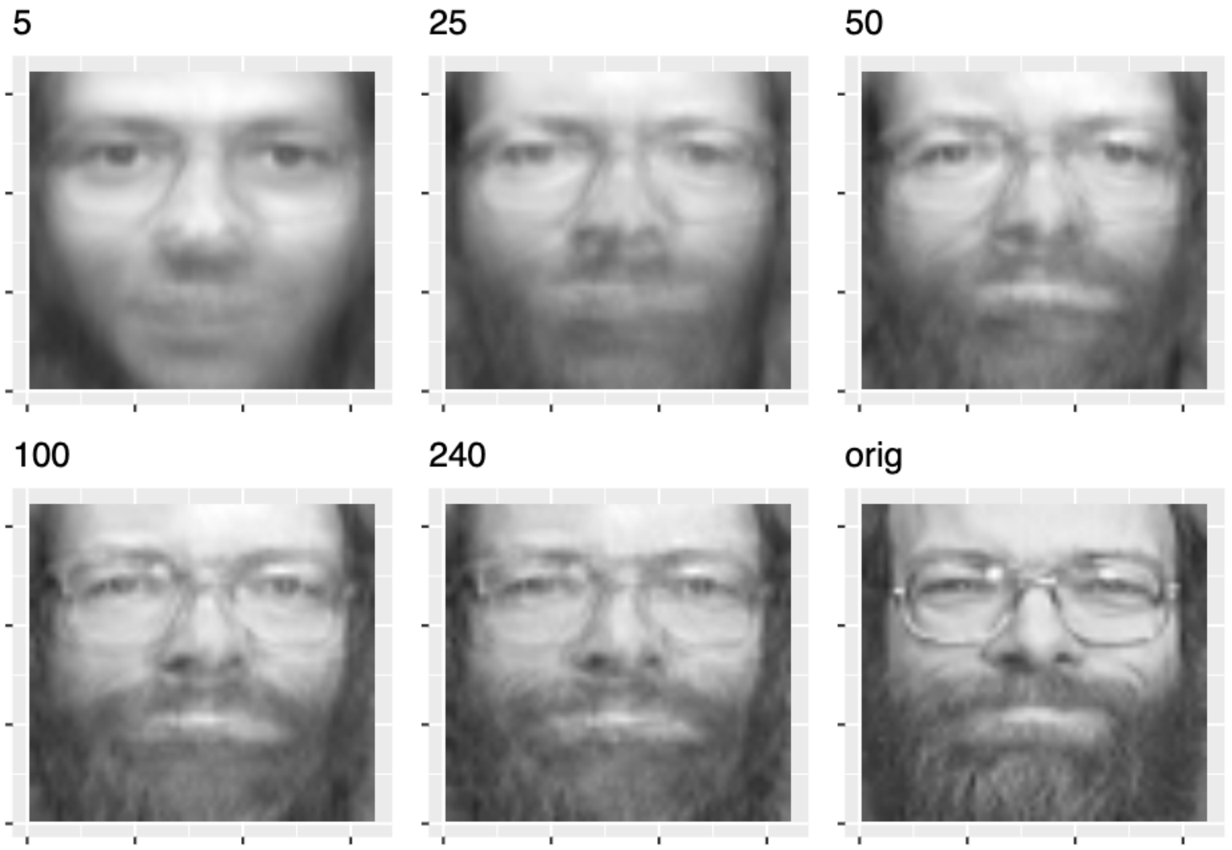
- *How well are any occluded parts recovered?*

```
# Read in data
F.df <- read.csv('~/MATP-4400/data/faces.csv')
# Save first column as labels
labels <- as.numeric(F.df[,1])
F.matrix<-as.matrix(F.df[,-1])
# Assign the row names
row.names(F.matrix)<-1:(nrow(F.matrix))

# A function to help plot faces
faceplot <- function(xx,width=64,midcolor="grey10",gcols=2,labels=row.names(xx)) {
 if (is.vector(xx)) {
   xx <- matrix(xx,nrow=1)
}
  pl <- vector("list",nrow(xx))
  for(i in 1:nrow(xx)){
```

```r
    face <- matrix(xx[i,], nrow=width)
    face.m <- melt(apply(face, 2, function(x) as.numeric(rev(x))))
    pl[[i]] <- ggplot(data=face.m) + geom_raster(aes(x=Var2, y=Var1, fill=value)) +
        theme(axis.text=element_blank(), axis.title=element_blank()) + guides(fill=FALSE) +
        scale_fill_gradient2(low="black", mid=midcolor, high="white") +ggtitle(labels[i])
  }
  grid.arrange(grobs=pl,ncol=gcols)
}


# Divide into trainface and  testface
# Since images occur in blocks of 10, work mod10 (using %%10) and get  images 4-9 as train and 0-3 as t
gg <- c(1:400)
h <- gg[ gg%%10 >3]
#h is the list of numbers of faces to extract
trainface.uncentered<- F.matrix[h,]
trainfacelabels<-labels[h]
# Test face
gg <- c(1:400)
h <- gg[ gg%%10 <= 3  ]
# h is the list of numbers of faces to extract
testface.uncentered<- F.matrix[h,]
testfacelabels<-labels[h]


# Find the mean face and plot it
ones<- matrix(1,ncol=1,nrow=nrow(trainface.uncentered))
train.mean <- 1/nrow(trainface.uncentered)* t(ones) %*% trainface.uncentered
# Mean center the data and save in trainface.
trainface<-trainface.uncentered-matrix(1,ncol=1,nrow=nrow(trainface.uncentered))%*%train.mean
# Principal component command. Do not recenter data.
my.pca<- prcomp(trainface,retx=TRUE,center=FALSE)


imagenum<-148
testimage<-testface.uncentered[imagenum,]
indices<-33:43
testmatrix<-matrix(testimage,nrow=64)
testmatrix[indices,indices]<-133 # This is our occlusion!
test.occluded <-matrix(testmatrix,nrow=1)
pl<-rbind(test.occluded,testface.uncentered[imagenum,])
faceplot(as.data.frame(pl),gcols=2,labels=c('occluded','original'))
```



```
tface<-148
# newface is the centered  test face
newface<-testface.uncentered[tface,]-train.mean
# Get the scalar projections of newface.
scalarproj <- newface %*% my.pca$rotation
recon<- t(my.pca$rotation)  # create the recon matrix as eigenvectors and then replace with reconstruct
# n is the number of eigenvectors found.
n<-nrow(recon)
recon[1,] <- train.mean+scalarproj[1]*recon[1, ]  #Reconstruct uisng the first eigenvector
# Add s_i v_i to each row.
for(j in 2:n){ recon[j, ] <- recon[j-1, ] + scalarproj[j]*recon[j, ]}
# Plot selected reconstructions followed by the original image
#test.occluded <-matrix(testmatrix,nrow=1)
pl <- rbind(recon[c(5,25,50,100,n), ])
pl <- rbind(pl,testface.uncentered[tface,])
# Plot the reconstructed faces for using 5,25,50, 100, the maximum eigenvetors, and the origial image
faceplot(as.data.frame(pl),gcols=3,labels=c('5','25','50','100',as.character(n),'orig'))
```



Exercise 3

- What are two ways that PCA can be used to improve a facial recognition system? (3 pts)

PCA helps reduce the amount of variables needed for facial recgonition system. The less variables needed, the easier it is to recognize a face which is a benefite of using PCAs. In addition, calculatin the covariance matrix is a lot easier and it is much quicker than other facial recognition systems.

Exercise 4

You may be curious about state-of-the-art facial recognition systems, FaceNet at Google in 2015. Read the following paper

FaceNet: A Unified Embedding for Face Recognition and Clustering

Answer the following questions

Facenet was a state-of-the-art in 2015 nearest-neighbor approach to image recognition. It uses deep-learning which is a machine learning method that is beyond the scope of this class. The rough idea is that the deep-learning methods is computing an embedding – i.e. a mapping to lower dimensional space – similar we do using PCA in Lab 6 and 7, but in a much fancier way. But we do know about training and testing accuracy from Lab 6 and 7. Read section Section 4 Datasets and Evaluation and Section 5. Experiments and look for how the data is prepared, how the training and testing sets are made, and how accuracy is evaluated. These questions are based on those sections.

1) How does Facenet prepare the image for recognition by default? Facenet utilizes the triple loss function on its images. It prepares the images by taking the batch, going through deep architecture, which is followed by L2 normilization, then goes through face embedding, and lastly followed by triple loss during training.

2) Facenet uses a hold-out or testing set. It divides this into pieces called splits so it can get a mean accuracy as well as a standard error. How many splits does it use? Facenet uses five splits.

3) Describe in your own words what true accepts measures and what false accepts measures. Why does facenet need to calculate both? False accepts measures the amount of times the facenet system accepted someone to have the same face but it was actually wrong and true accepts measures the amount of times the facent system accepted someone to have the same face and it was correcr. Facenet calculates both because it is important to measure how many times an error occurs with the software and how many times an error does not occur.

4) How accurate as measured by VAL is the system at JPEG quality 50? JPEG quality 50 is 85.5% accurate as measured by VAL.

5) How many bytes does Facenet recommend using for it system? Why did it pick this number? Facenet recommends using 128bytes for its system because this is the number that the data can be quantized without loosing accuracy.

6) Facial recognition can be be mis-used causing ethical concerns. Describe one potential misuse of the data and what ethical concerns are associated with this use? One potential misuse of facial recognition is when organizations save images and pictures of random people, specifically, without their consent. Facial recognition has brought up the debate about privacy and whether innocent civilians images should be saved and stored in a database because these images can end up being used against them. For example, in the case of false accepts if facial recognition systems incorrecelty identify a person that can lead to issues because an innocent person is being accused while the guilty is left off. This creates the debate of whether people shoud continute giving orginzation access to their profiles.

Extra Credit

Data analytics is playing an important role in understanding the COVID-19 pandemic.
Below is a graph showing how the cummulative number of deaths per state are evolving through time. Your job is change this graph to a more effective visualization that illustrates the evolution of the pandemic in the United States. This coud be a single visualization or a set of visualizations. One possible idea is to use published graphs such as this one as an inspiration https://www.kdnuggets.com/2020/04/top-tweets-apr25-31.html. But design your own visualization involving one or more graphs. Be creative to get full points.

We have a provided a sample graph showing the number of COVID-19 deaths by state upto April 2.

Read in the data

```
# Read LONG dataframe from file system
# used read_csv from tidyverse because it automatically converts second column into dates
covid_TS_states_long<-read_csv('~/MATP-4400/data/covid_TS_states_long.csv')
```

```
## Parsed with column specification:
## cols(
##   NAME = col_character(),
##   date = col_datetime(format = ""),
##   deaths = col_double()
## )
```

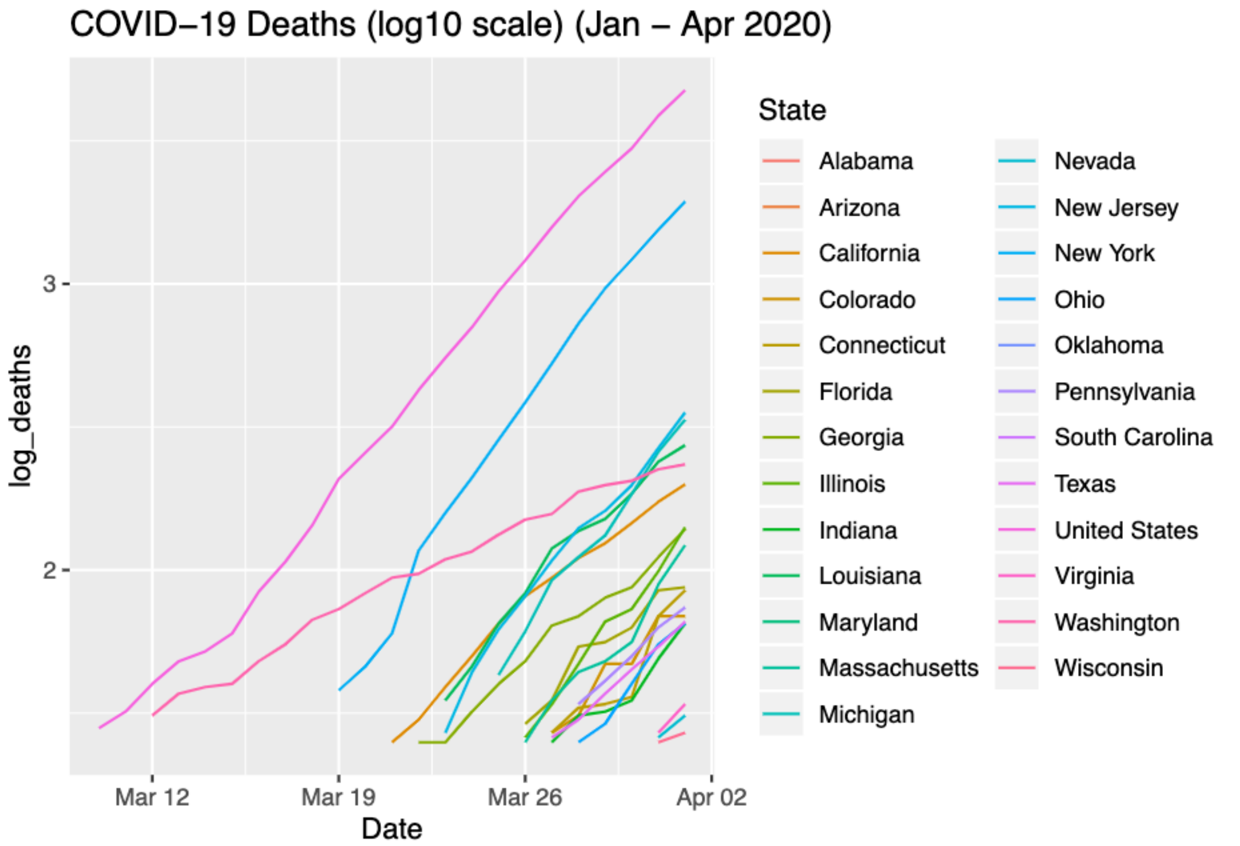
For the sample plot, we transformed the data. We only showed data for states with more than 25 deaths. The deaths are put on a log scale. The data is

```
# Prepare data: group by data
 covid_TS_states_long <- covid_TS_states_long %>%
   filter(deaths >= 25)
# convert to table with columns  data into table with NAME, data, deaths,log_deaths
covid_TS_plot <- covid_TS_states_long %>%
  group_by(date)
```

```
covid_TS_plot$log_deaths <- log10(covid_TS_plot$deaths)
# make the state name a factor
covid_TS_plot$NAME <- factor(covid_TS_plot$NAME)
```

Draws the sample plot. It could be much improved.

```
p.log <- covid_TS_plot %>%
  mutate(
    State = NAME,      # use NAME to define separate curves
    Date = update(date, year = 1)  # use a constant year for the x-axis
  ) %>%
  ggplot(aes(Date, log_deaths, color = State)) +
  geom_line() +
  ggtitle("COVID-19 Deaths (log10 scale) (Jan - Apr 2020)")

p.log
```



In the sample plot, we see that the US and most states appear. If they appear as almost as straight line, this means that the number of deaths is increasing exponentially. Lines with greater slope indicate that the exponential rate of increase in deaths is greater. The line will become horizontal when there are no more COVID-19 deaths.

Discuss what your visualizations indicate about the spread of COVID-19 in the United States. Talk about the differences that you see between states and what that may mean now and as the pandemic evolves. 3 points extra credit.