Exercise 2

1. [0.5 points]

We read the data into out environment and have a look at the first 10 pixelvalues of the first ten digits.

```
digits_sample <- read.csv2(paste0("C:/Users/leaz9/OneDrive/Dokumente/StatLearn WS22/", "/data/digits.cs
dim(digits_sample)
```

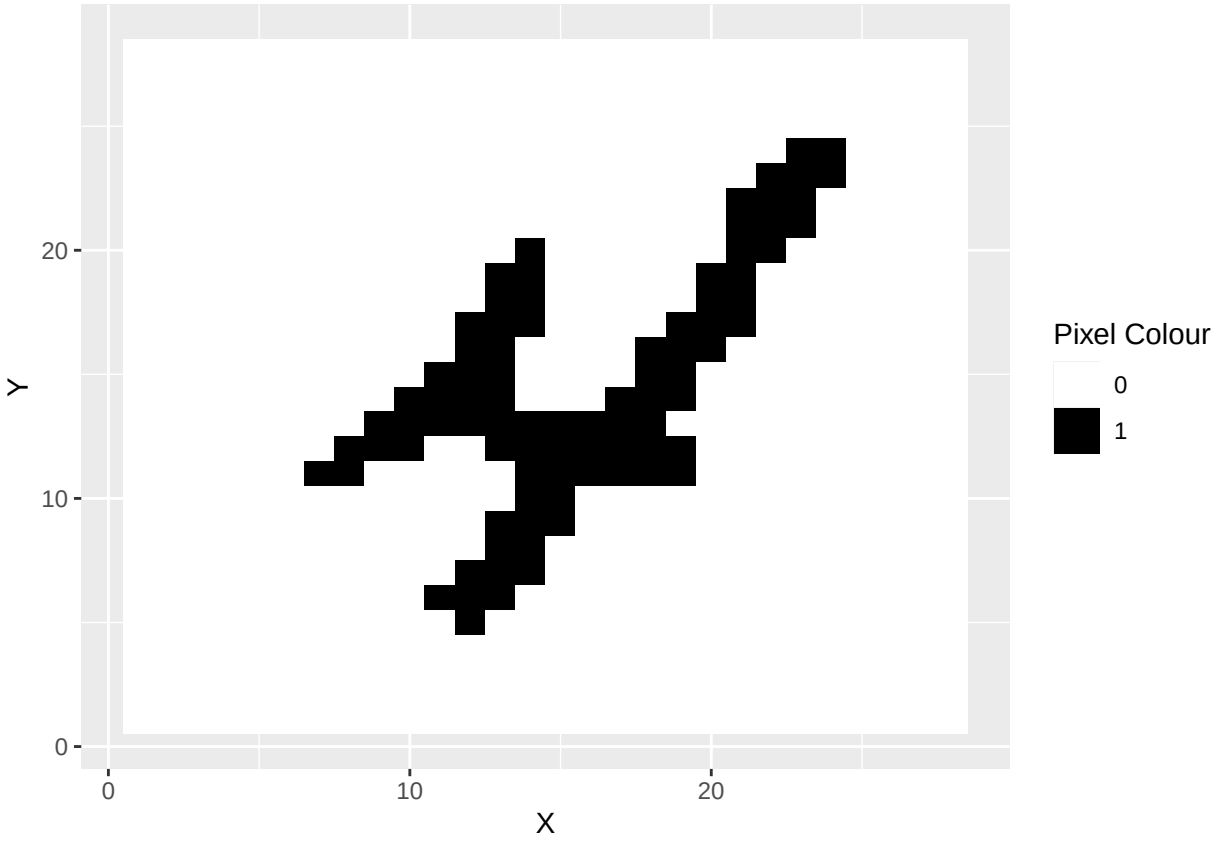
```
## [1] 5000  784
```

```
digits_sample[1:10, 1:10]
```

```
##    V2 V3 V4 V5 V6 V7 V8 V9 V10 V11
## 1   0  0  0  0  0  0  0  0   0   0
## 2   0  0  0  0  0  0  0  0   0   0
## 3   0  0  0  0  0  0  0  0   0   0
## 4   0  0  0  0  0  0  0  0   0   0
## 5   0  0  0  0  0  0  0  0   0   0
## 6   0  0  0  0  0  0  0  0   0   0
## 7   0  0  0  0  0  0  0  0   0   0
## 8   0  0  0  0  0  0  0  0   0   0
## 9   0  0  0  0  0  0  0  0   0   0
## 10  0  0  0  0  0  0  0  0   0   0
```

As stated in the text, the 28×28 matrix that represents an image was flattened rowwise, so when we assemble the dataframe to plot the first observation, the first 28 pixel values have position $y = 28$ and x from 1 to 28, the next 28 values have position $y = 27$ and x from 1 to 28 and so on.

```
library(ggplot2)

df1 <- data.frame( X = rep(1:28, 28), Y = rep(28:1, each = 28),
                   PixelCol = as.numeric(digits_sample[1, ]))

ggplot(df1, aes( x = X, y = Y, fill = as.factor(PixelCol))) +
  geom_tile() +
  scale_fill_manual(values = c("white", "black") )+
  labs( fill = "Pixel Colour")
```



This looks like a 4.

2. [0.5 points]

```
library(flexmix)
```

```
## Loading required package: lattice
```

```
nc <- 5
set.seed(1)
bmm <- flexmix(as.matrix(digits_sample) ~ 1,
                               k = nc,
                               model = FLXMCmvbinary(),
                               control = list(iter = 100))

bmm
```

```
##
## Call:
## flexmix(formula = as.matrix(digits_sample) ~ 1, k = nc, model = FLXMCmvbinary(),
##     control = list(iter = 100))
##
## Cluster sizes:
##    1    2    3    4    5
##  981  892 1381  871  875
##
## convergence after 56 iterations
```

3. [1 point]

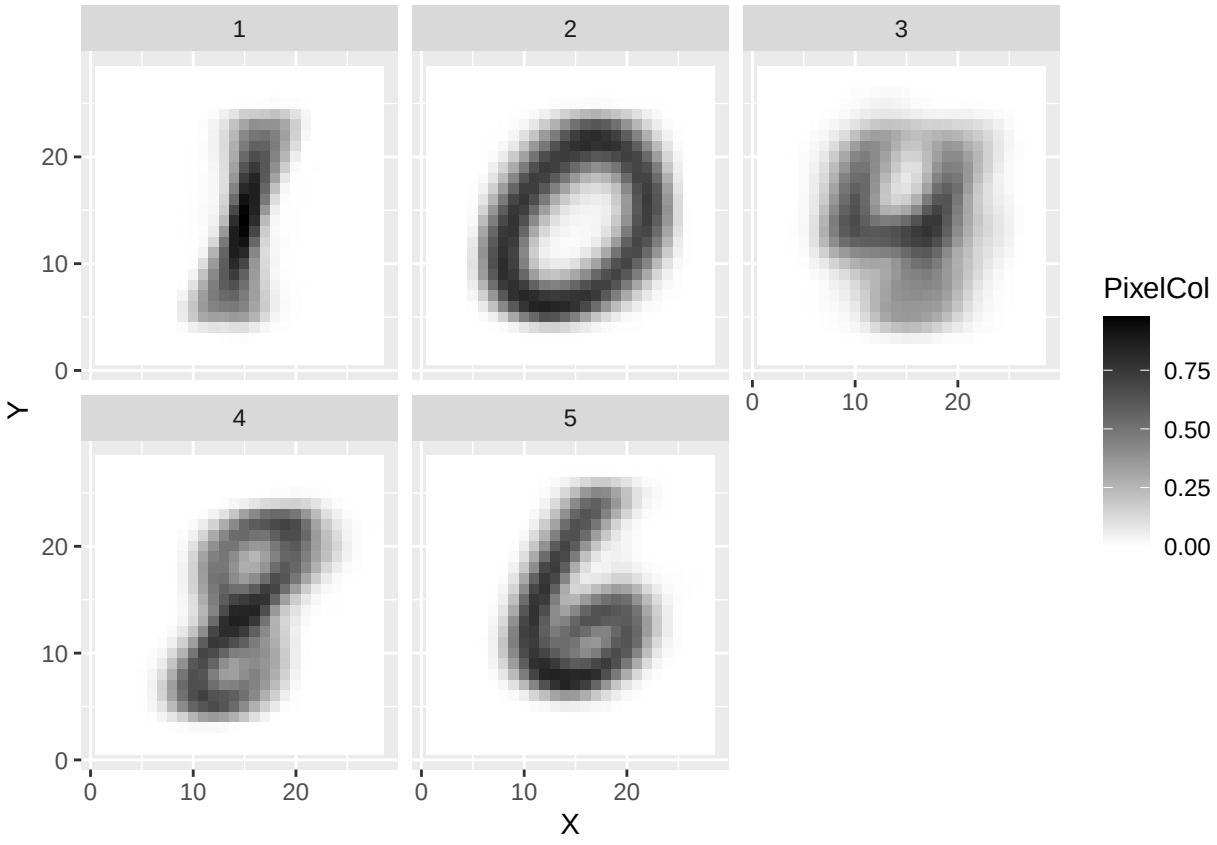
First, we save the estimated parameters of the model.

```
params <- parameters(bmm)
dim(params)
```

```
## [1] 784   5
```

For all 5 clusters we obtain a vector $(\pi_{1,g}, \pi_{2,g}, \ldots, \pi_{784,g})'$ giving the probability that a pixel is black or white. Therefore `params` is of dimension 784×5. To assemble a long dataframe, we transform it to a vector (the function `as.vector()` flattens the matrix columnwise) and store the probabilities as pixelvalues. These values between 0 and 1 can then be plotted in greyscale. For the pixel positions, we repeat the structure that we have found in 1 (repeat it 5 times for the 5 clusters). The values 1 to 5 in the column `Cluster`, that specify the cluster number, are each repeated 784 for the 784 pixel values.

```
params_vec <- as.vector(params)

df <- data.frame( PixelCol = params_vec, Cluster = rep(1:nc, each = 784),
                  X = rep(rep(1:28, 28),nc),  Y = rep(rep(28:1, each = 28), nc))

ggplot(df, aes( x= X, y = Y, fill = PixelCol))+
  geom_tile()+
   scale_fill_gradient(low = "white", high = "black")+
   facet_wrap(~ Cluster)
```



Looks like we have samples from the digits 0,1,4,6,8! You can save the plot with the function `ggsave()`.

4. [1 point]

First, we read the new observation into our environment. We see that it is a dataframe with only one column named `x` with the 784 pixel values.

```
new_dig <- read.csv2(paste0("C:/Users/leaz9/OneDrive/Dokumente/StatLearn WS22", "/data/new_digit.csv"))

head(new_dig)
```

```
##   x
## 1 0
## 2 0
## 3 0
## 4 0
## 5 0
## 6 0
```

Now we save the cluster probabilities to a vector named `prob_cl`. We need to compute the conditional probabilities

$$\Pr(\mathbf{X} = \mathbf{x}|G = g) = \prod_{j=1}^{784} \pi_{j,g}^{x_j}(1 - \pi_{j,g})^{1-x_j}$$

for the new observation $\mathbf{x} = (x_j)_j$ and all 5 clusters. The function `dbinom()` is the probability mass function of the Binomial distribution. It can handle vectors as input, for x as well as for p. The pmf is then evaluated at the pairs $(x_j, p_j)_j$, which is exactly what we need.

```
prob_cl <- prior(bmm)

cond_x <- apply(params, 2,
                    function(p){prod(dbinom( x = new_dig$x, size = 1,  prob = p ) )} )

cond_x
```

```
##        Comp.1        Comp.2        Comp.3        Comp.4        Comp.5
##  0.000000e+00 1.215969e-127 1.945976e-104  1.680909e-69  0.000000e+00
```

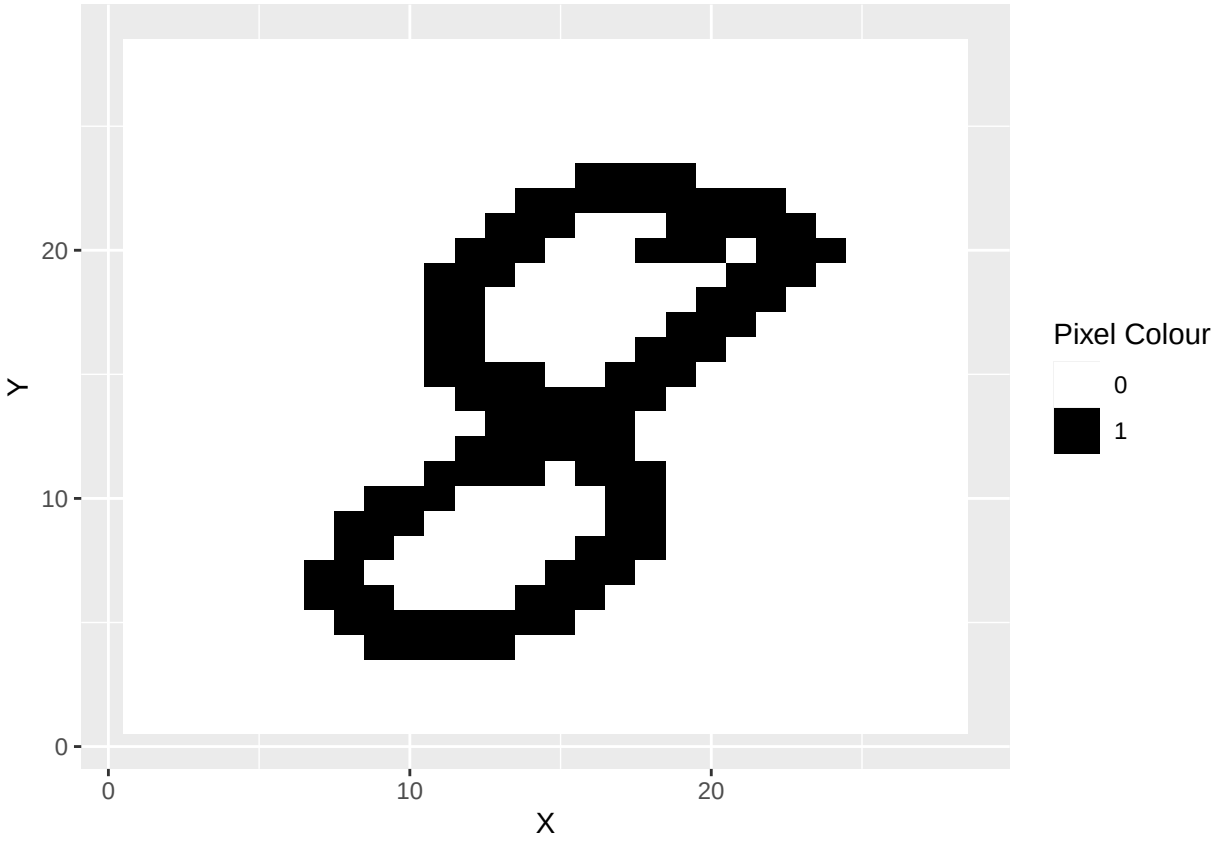
To compute $\Pr(G = g|\mathbf{X} = \mathbf{x})$ for each clutser, we use formula (5.1) from the lecture, so we need to multiply the conditional probabilites above with the prior probabily of the respective cluster, and then divide by the sum over these terms.

```
probs <- cond_x*prob_cl

probs/sum(probs)
```

```
##       Comp.1       Comp.2       Comp.3       Comp.4       Comp.5
## 0.000000e+00 7.413864e-59 1.828342e-35 1.000000e+00 0.000000e+00
```

With a probability of almost 1 we the new observation belongs to cluster number 4, i.e. it is almost surely an 8. Let's plot it and see.

```
dfnew <- data.frame( X = rep(1:28, 28), Y = rep(28:1, each = 28),
                     PixelCol = new_dig$x)

ggplot(dfnew, aes( x = X, y = Y, fill = as.factor(PixelCol))) +
  geom_tile() +
  scale_fill_manual(values = c("white", "black") )+
  labs( fill = "Pixel Colour")
```



It really is an 8.

5. [0.5 points]

We load the true class labels and recode the cluster numbers to the numbers they represent (for improved readability of the contingency table). Then we compute the contingency table with the function `table()`.

```
true_labels <- read.csv2(paste0("C:/Users/leaz9/OneDrive/Dokumente/StatLearn WS22", "/data/true_labels.

cluster_est <- bmm@cluster

cluster_est <- dplyr::recode(cluster_est, "1" = "1", "2" = "0", "3" = "4", "4" = "8", "5" = "6")

table(cluster_est, true_labels$x)
```

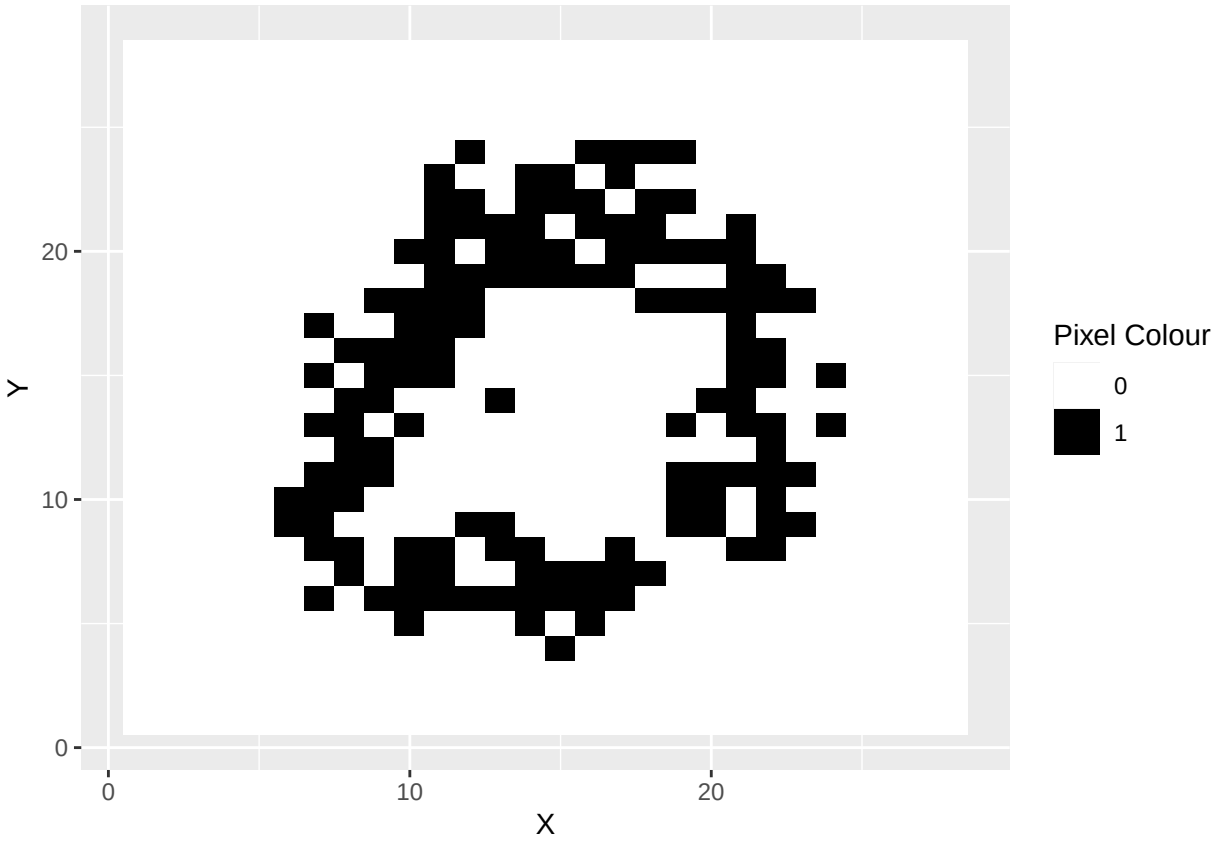
```
##
## cluster_est   0   1   4   6   8
##           0 858   0   0  17  17
##           1   0 924  12  24  21
##           4  39  16 897 144 285
##           6  51   7  12 790  15
##           8  38 166  48  27 592
```

It's not too bad, but there are for example 285 out of 930 eights that are classified as fours.

6. [0.5 points]

We take the parameter vector $(\pi_{j,2})_j$ of cluster 2 and sample one observation for each $\pi_{j,2}$. Then we assemble a dataframe as before and plot the sampled observation.

```
set.seed(2021)

sample_cl2 <- sapply(as.vector(params[, 2]),
                     function(p){rbinom(n = 1, size = 1, prob = p)})

df_samp <- data.frame( X = rep(1:28, 28),
                       Y = rep(28:1, each = 28), Colour = sample_cl2)



ggplot(df_samp, aes( x = X, y = Y, fill = as.factor(Colour))) +
  geom_tile() +
  scale_fill_manual(values = c("white", "black") )+
  labs( fill = "Pixel Colour")
```



Indeed, this looks a bit like a zero (cluster 2 represents the zeros). However, the simplified assumption of independence of the pixels results in this perforated appearance.