Notes:

- In Session 1.1 (today afternoon), please do at least Tasks 1 and 2. Try to do Task 3 if there is time.
- Some of the Tasks will require consultation of the provided Handout.
- Full course material is available at[1]

  `http://www.maths.dur.ac.uk/~dma0je/PG/Mix/`

- At the link given above you will also find solutions (R code), but please only use this if you are *really* stuck!
- If you have not attended Session 1.1, then please carry out quickly Tasks 1(a-d,h), 2(h), 3(a), and then start with Task 4.

Task 1: Preliminaries: Working directory, data frames, and workspace

(a) Every R session uses a *working directory*. This is some directory on your hard drive (or USB key, etc.) which R will use by default in order to save images, data, or your workspace (see item (g)). R will also assume by default that any data sets that you attempt to read in are stored in this directory. Please create an empty directory with the name *Mixtures* somewhere (remember where!), which we will use as working directory for this course.

(b) Download the data set *energy.csv* from the link given above into the directory created in part (a). A description of these data is given in the handout.

(c) Open R. Check your current working directory via

```
getwd()
```

and then use

`setwd("pathname")`

where "*pathname*" is the path name of the directory chosen in (a), in order to tell R that this shall serve as your working directory from now on.

(d) Read in the data set *energy.csv* into an object named `energy.use`, using the instruction given in the handout.

(e) Check whether things have gone right. Try

```
dim(energy.use),
```

this should give you the dimension (135×52) of the data matrix. Also try

[1] Note that the '0' in `dma0je` is a 'zero'

```
head(energy.use)
```

in order to see the first 6 rows.

(f) The object `energy.use` is a *data frame*. You can check whether or not an object is a data frame by typing

```
class(object)
```

or

```
is.data.frame(object)
```

Try this for the object `energy.use`. It is easy to access individual rows, columns, or elements of a data frame. For instance,

```
energy.use[32,]
```
and
```
energy.use[,49]
```

which will give you the 32nd row and 49th column, respectively, and

```
energy.use[32,49]
```

will give you the entry of the 32nd row and the 49th column (this is the Cyprus energy consumption in 2007). You can also access columns directly through their column names, such as

```
energy.use$X2007 .
```

Data frames are very important as they are the standard form in which data are expected by many R functions, such as `lm`, `glm`....

(g) Another important concept is the *workspace*, which contains all objects that you create during an R session. For instance, type `x<- 1` and then

```
ls()
```

This will list all objects contained currently in your workspace (in your case, `energy.use` and `x`). You can, at any time, save your entire workspace for later use, by using the command `save.image`("*filename*"). Let's do this. Type

```
save.image("mixture.RData"),
```

then close R and open it again. Then load the saved workspace back via

```
load("mixture.RData"),
```

and type `ls()` again to check whether everything is there!

(h) Finally, let us simplify the data frame a little bit, so that it is easier to use for the applied work. We reduce our interest to the energy consumption in the years 2001 and 2007. We do this via

```
energy <- energy.use[,c("X2001", "X2007")]
```

Also, we would like to give the rows and columns of the new data frame meaningful names. Please type

```
rownames(energy)<- energy.use[, 1]
colnames(energy)<-c("use01", "use07")
```

in order to specify row and column names, respectively. Then type `energy` to look at your final data frame. This data frame allows to access information quickly. For instance,

```
energy["Cyprus",]
```

gives you the Cyprus values of energy consumption. Do this for a couple of countries.

Task 2: Basic programming and operations with data frames

Note: As the code sequences are becoming longer now, it would be useful to use an editor from now on for your working. Depending on your system, good choices are RStudio, Tinn-R, or emacs.

(a) One may be interested in looking at these data in a form that they are ordered by their energy consumption. This can easily be done using

```
order(energy$use07)
```

which gives you a list of numbers. The first number tells you the index (here: 39) of the country with the smallest per-capita energy consumption (here: Eritrea), and typing

```
energy[order(energy$use07),]
```

gives you the full ordered list.

Save this ordered data frame into a new data frame `senergy`.

(b) Next, we wish to identify the nations with extremely large energy consumption, say, more than 10000 kg of oil per capita (Intuitively, what do you think, which countries will this be?). Typing

```
energy$use07 > 10000
```

will give you a vector of logical values, with a `TRUE` for each country for which this condition is met. The command

```
sum(energy$use07 > 10000)
```

will tell you how many these are, and

```
which(energy$use07 > 10000)
```

will give you the index numbers of these countries. From this, we would get the data rows corresponding to these countries via

```
energy[which(energy$use07 > 10000),]
```

(c) We would like to compare the energy use in 2001 and 2007. Do the same as in (b) but now use the condition `energy$use01 > energy$use07` instead. Observe and understand the information that you gain at each step.

(d) A very useful tool to carry out repeated operations is the `for` command (see Handout!). Task: Implement a loop which, for all 135 countries, writes a text like

> In 2007, the energy use in *country* was equivalent to *value* kg oil per capita.

(e) Another command for repeated operations is `while`. It does not have a fixed number of loops, but proceeds until a certain condition is met. For instance, consider the ordered list of countries created in (a). Assume we are interested in the following question: If we take exactly one person from each of the countries with the smallest energy use, i.e. one person from Eritrea, one person from Bangladesh, etc., then how many persons are needed in order to achieve the same use of energy as a single person in Qatar?

To answer this, create objects `i` and `sum07` and assign them the initial value 0. Then use the `while` function (see handout) with *condition*

```
sum07< senergy["Qatar",2]
```

and *action*

```
i     <- i+1
sum07 <- sum07+  senergy[i,2]
```

Make it clear to yourself what each row does. Also, interpret the result.

(f) Use `apply` to compute a vector which contains, for each country, the larger of the two energy consumption values given for 2001 and 2007. Consult the handout and the corresponding help file (via `help(apply)`) if you are unsure how to do this.

(g) Use `hist` and `boxplot` to create histograms and boxplots of the variables `use01` and `use07`. Comment on the distributional shape.

(h) Next, add logarithmic versions of these variables, say say `luse01` and `luse07`, to the data frame via

```
energy$luse01<- log(energy$use01),
```

and for `use07` analogously. Repeat question (g) using the transformed variables. What can we say about the distribution of these transformed variables, compared to the original ones?

Task 3: Motivation and theory for finite Gaussian mixtures

(a) Look again at your histogram from the `luse01` variable. One may be interested in *modelling* this distribution, i.e. describing it through some parametric family of distributions. A Gaussian distribution appears to be inadequate due to the clear dip in the center. But, perhaps one could argue that *two* Gaussian distributions are involved here: One Gaussian centered at about 6.5, and another one centered at about 8.5, and both of them roughly equally likely. This leads to the idea of mixture modelling, to which we turn in this section.

Next, we consider a data set featuring $n = 82$ observations of galaxy velocities. Load the `galaxies` data (see handout), read the associated help file, and create a histogram using the option `breaks =18` in function `hist`.

The histogram clearly tells that there are several modes (perhaps, 4 or 5) present in the data, each of them associated with strongly differing proportions of the total sample size. These different modes may be evidence for the presence of some unobserved heterogeneity, for instance due to the size or distance of the galaxies.

Here again, one could think of the distribution of the data as an overlay of several Gaussian density functions, but now with unequal probabilities. The general term for such a situation is a "Finite Gaussian mixture". See the Handout for details.

(b) If you have arrived at this point before the lecture, then please take a paper and pencil, and attempt to derive equations (3), (4), and (5) on the Handout. You can do this by taking the appropriate derivatives of equation (8). Begin with the μ_k.

Notes:

- *Please attend the lecture* **"EM algorithm for finite Gaussian mixtures"** *in Session 1.2 before you proceed further.*
- *If you leave the computer now, don't forget to* **save your workspace** *as explained in 1(g).*

Task 4: Implementing the EM algorithm for Gaussian mixtures

(a) Implement now the EM-algorithm for finite Gaussian mixtures. You will need for this

– a function which implements the E-step. The framework for this function is

```
estep <- function(dat,  pi, mu, sigma){
  n <- length(dat)
  K <- ...
  W <- ...
  for (i in 1:n){
       W[i,]<- pi* exp(...)/sum(...)
     }
  }
 return(W)
}
```

This function takes the data $y =$ `dat` and the current parameter estimates $\hat{\theta} =$ (`pi`, `mu`, `sigma`) as arguments, and produces the matrix `W` $= (w_{ik})_{1\leq i\leq n, 1\leq k\leq K}$. Note that `pi` and `mu` are vectors, and `sigma` is a scalar.

– another function which implements the M- Step:

```
mstep <- function(dat, W){
   n  <- dim(W)[1]
   K  <- ....
   pi <- ....
   mu <- apply(W*dat,2,sum)/apply(W,2,sum)
   ...
   sigma <-

   return(list("pi"=pi, "mu"=mu, "sigma"=sigma))
}
```

Note: There are several ways to implement the estimate of σ, but without advanced programming expertise you will probably need some iterative way of doing this. For instance, one can loop over i and K and add the relevant term at each stage. Or, one can firstly construct a $n \times K$ matrix, say `diff`, of squared differences $(y_i - \mu_k)^2$, and then use `sum(W*diff)`.

– a third function which makes the actual EM-algorithm; something like

```
em <- function(dat, K){
   # EM Starting values:
   # (Be creative: what could be sensible generic starting values,
   # in the absence of the knowledge of the true mixture parameters?)
 pi    <- ...
 mu    <- ...
 sigma <- ...
  # EM iterations:
 while (...){     # define a simple stopping or convergence criterion
     W   <- estep(dat, pi, mu, sigma)
     fit <- mstep(dat, W)
```

```
    pi  <- fit$pi
    ...
  }

  fit <- list( "pi"=pi, "mu"=mu, "sigma"=sigma, "W" =W )
  return(fit)

}
```

(b) Apply your implemented algorithm on the `luse01` and `galaxies` data. Attempt to visualize the results. Do they look reasonable?

Task 5: Simulation and Bootstrap

(*Note: Advanced and a bit challenging – you can skip this part if you have less than, say, 20 minutes left at this stage.*)

(a) Read the section on Simulation given on the Handout, then complete the missing code line below to implement this simulator:

```
gauss.mix.sim1 <- function(K, pi, mu, sigma){
   x   <-runif(1)
   sim <- 0
   cpi <- cumsum(pi)
   k   <-1
   while (x>cpi[k]){
           k<-k+1
   }
   sim <-  ??????

   return(sim)
 }
```

`gauss.mix.sim1` is a function which simulates *one* observation. Extend the code and write a function `gauss.mix.sim` which simulates an arbitrary number, say `n`, of replicates. Try your function using a value of θ of your choice, plot the histogram, and observe whether the result looks right.

(b) Improve your function `em` so that it returns the likelihood and the disparity as additional outputs. Read the instructions in the handout before your attempt this. Then, implement the likelihood ratio test for testing for the number of mixture components K, following the instructions provided on the handout.

Task 6: Multivariate extension

For multivariate data, nothing is much different (see last page of handout). We do not have time to attempt a full implementation, therefore we use an existing R package, which implements exactly this methodology.

(a) Download the R package **UEM** from the link given on the first page of this task sheet (Use the .ZIP file for Windows or Mac, and the .tar.gz version under Linux. The module leader can provide you with the file if you can't connect to the internet. Then, install the package (Depending on the OS and your R environment, there are different ways of doing this. In R for Windows, the easiest option is to use the menu item 'Install package(s) from local zip files'). Then, load the package via `library(UEM)`.

(b) Type `data(rock)`, `head(rock)`, `?rock` and read the help file. We are interested in the `peri` and `shape` variables from now on. Hence, use `rock23 <- rock[,2:3]` to obtain a reduced data frame, and display it via `plot(rock23)`.

(c) Now, we fit a bivariate Gaussian mixture to this data set. Use

```
fit2 <- EM(rock23,K=2, plot=TRUE)
```

to do this. Type `names(fit2)` which gives you the collection of components of the fitted mixture. Look at some of these, and try to match them to the expressions given on the handout. Specifically, what is it what you see when typing `fit2$var`?

(d) Let's look at the matrix of posterior probabilities. Type

```
round(fit2$W, digits=3)
```

You see from this that all but two of the observations are perfectly allocated to a cluster center. Looking at the image created in part (c), which ones (do you think) are these two observations? Then type

```
identify(rock23$peri, rock23$shape)
```

to check, by clicking on individual observations, whether you have been right!

(e) If there is still time, continue to play around with changing the number of mixture components, etc.