7. Software for fitting Multimix

7.1. R templates for fitting Multimix

I find the easiest way to fit `Multimix` models is by using a template. A template is not a run-able R script, but a mixture of R code with comments indicating how to complete it. Here is an example:

```
# Use setwd(folder) to change to a folder in which the \verb|Multimix|
    scripts are.
setwd("[1.]")

# Global constants which might be altered between different
# runs on the same data.
qq <-   "[2.]"    # Number of clusters
# [using "q" would conflict with R's quit function]
nIt <-  "[3.]"    # Maximum number of iterations
eps <-  "[4.]"    # Minimum increase in loglikelihood per EM step.
# [stopping criterion]
cdep  <- "[5.]" # list of MVN cells. [May be NULL.]
lcdep <- "[6.]" # list of location cells,
# discrete first in each cell. Also may be NULL.
"[7.]" # Note: Once the data frame has been set up, check that the
# values of cdep and lcdep model the correct variable associations
# for the model you are fitting.

library(mvtnorm)


"[8.]" # Load Multimix scripts (mostly function definitions)

source("Functions.r")
source("data_organise.r")
source("make_Z_discrete.r")
source("make_Z_random.r")
source("make_Z_fortran.r")
source("first.Z.to.P.r")
source("P.to.Z.r")
source("Z.to.P.r")
source("mmain2.r")

# Read data csv or txt into raw dataframe

rawdf <- read.csv("[9.]", header=)

"[10.]" # Make function converting raw dataframe into Multimix-ready
    dataframe.
```

```
# Categorical variables should be changed to factors with appropriate
    labels.
# Continuous variables may need to be scaled or transformed.

construct <- function(rawdf){

      #(data-dependent function body)
}


# Apply function to raw dataframe.

dframe <- construct(rawdf)

# Save dimensions of 'dframe'.

n <- dim(dframe)[1]
v <- dim(dframe)[2]

# Build list of needed functions of the data

D <- data_organise(dframe, cdep, lcdep, minpstar = 1e-9)

"[11.]" # Build starting values for Z matrix and parameter list P.

      #(Use "make_Z_random.r" or similar script to construct a starting Z
     matrix, then use "first.Z.to.P.r" to construct P.)

      #(Alternatively, if starting from a parameter list P, use the
    script "P.to.Z.r" to construct the initial Z.)

# Now starting values of Z and P should have been constructed.

"[12.]" # Apply mmain() function

zpr <- mmain2(D,Z,P)

"[13.]" # Trim results matrix.

ok <- apply(zpr$results[,1:qq + 1],1,sum) > 0.5
zpr$results <- zpr$results[ok,]
tail(zpr$results)
```

The above template listing includes positions designated by "[1.]", "[2.]",... What needs to be added at each position is described in the following list:

1. Give the pathname for the working directory in a string here. This should contain all the R scripts listed in Appendix C.

2. An integer giving the number of components being fitted, for example 4.

3. An integer giving the maximum number of iterations to be performed, for example 2000. The reason for specifying this number is to ensure that the algorithm ter-

minates. This may be difficult to specify on the first run with a new data/model setup, but becomes clear after a few runs.

4. A numeric value giving the minimum increase in log-likelihood required for the iteration steps to continue. I have tended to use the values 1e-9 or 1e-10. In an EM a logarithm the log-likelihood must increase at every step, but after a while there are diminishing returns.

5. A list of m-tuples of column numbers in the data frame, $m \geq 2$. The numbers should correspond to attributes with continuous values. Example: in the Cancer example described below columns 5 and 6 contain values for the systolic and diastolic blood pressures of the patients. Entering *list(c(5,6))* will cause the blood pressure variables to be modelled by bivariate normal distributions within each cluster, and no other multivariate normal distributions will be included in the model.

6. A list of m-tuples of column numbers in the data frame, $m \geq 2$. In each m-tuple the first number corresponds to a categorical attribute. Example: *list(c(12, 2, 8), c(3,1))* . Again this comes from the Cancer example. Column 12 is the binary attribute *Bone metastases* and column 3 is the ternary attribute *Performance*. The other column numbers refer to continuous variables. Each list member specify an Olkin-Tate Location model for its columns within each cluster.

7. Note: Once the data frame has been set up, check that the values of cdep and lcdep model the correct variable associations for the model you are fitting.

8. Load `Multimix` scripts (mostly function definitions) from the working directory.

9. *Name of data file in working directory.* Read data csv or txt into raw dataframe using **read.csv** or **read.table** as appropriate. Use TRUE or FALSE for the value of *header* as appropriate.

10. Make function converting raw dataframe into Multimix-ready dataframe. Categorical variables should be changed to factors with appropriate labels. Continuous variables may need to be scaled or transformed. Refer to examples below for help in writing the body of the function *construct*.

11. Use one of the scripts defined in Appendix C.3.3 to make an initial value of the matrix Z.

12. The function **mmain2** is defined in Appendix C.3.6.

13. The logical variable *ok* checks that the sum of the class probabilities, which should be 1 at each iteration, is greater than 0.5.

7.2. Example of fitting Multimix models in R: Prostate Cancer Data

We use R to fit three models to the dataset described in Appendix A.1, the Byar-Green Prostate Cancer Data.

The algorithm that we use is an iterative algorithm that is a special case of the EM algorithm for maximum likelihood estimation. As such there are three things that we need to specify.

1. We need to specify the model that we are fitting. There are a number of choices for a `Multimix` model:

   local independence Within each cluster the variables are distributed independently. If the variable is continuous it is given a normal distribution. If the variable is categorical it is given a discrete distribution.

   weakened local independence The variables are partitioned into groups called **cells**. The cells with a single variable are distributed as in the local independence case. The cells with two or more continuous variables are given a multivariate normal distribution. The cells with two or more variables containing exactly 1 categorical variable and one or more continuous variables are given an Olkin-Tate distribution.

2. We need to give **starting values** to the model parameters and cluster probabilities. Alternatively we may supply positive starting values to the elements z_{ij} of the $n \times q$ matrix Z, where z_{ij} is the probability that observation i belongs to cluster j.

3. We also need to ensure that the algorithm terminates, perhaps by specifying a maximum number of iterations.

7.2.1. Working Directory

We will assume that the Working Directory is as listed below.

```
C:\Files\Writing\Multimix F & R\Template\CancerT>dir
 Volume in drive C is Windows
 Volume Serial Number is 7EF8-AD36

 Directory of C:\Files\Writing\Multimix F & R\Template\CancerT

27/09/2022  10:12 pm    <DIR>          BP
25/09/2022  04:25 pm            22,624 Cancer RvF comparison.txt
10/11/2017  07:24 pm            27,075 CANCER11a.DAT
22/09/2022  04:06 pm             3,111 CancerT.r
21/05/2022  01:07 pm             3,753 data_organise.r
23/05/2022  04:26 pm             4,143 first.Z.to.P.r
21/02/2019  03:18 pm               312 Functions.r
```

```
27/09/2022  10:38 pm    <DIR>          LI
25/07/2021  06:25 pm               396 make-Z-from-Stage.r
24/05/2022  11:21 am               281 make_Z_discrete.r
26/09/2022  09:40 pm               242 make_Z_fortran.r
26/09/2022  05:01 pm               231 make_Z_groups_out.r
20/05/2022  05:22 pm               344 make_Z_random.r
08/09/2022  10:12 pm            26,726 multimix90.f90
21/05/2022  01:34 pm               514 mmain2.r
27/09/2022  09:02 pm    <DIR>          OT
21/05/2022  01:45 pm             2,505 P.to.Z.r
29/01/2019  11:42 pm               958 Stage.txt
21/05/2022  01:56 pm             3,489 Z.to.P.r
```

You will notice that there are Fortran-related files in this directory, this is because we will be using this same directory later to fit `Multimix` using the Fortran version. The file `Stage.txt` contains a classification of the 475 patients into stage 3 or stage 4 Cancer. We will not treat this as a supervised learning problem, but rather seek to cluster the data into 2 clusters based on the attributes alone. In other words we contemplate that the stage information may possibly be erroneous.

7.2.2. Editing the R template to make the script to fit the local independence model to the Prostate Cancer data

The numbers in the numbered list below correspond to the numbers of form `"[n.]"` in the template.

1. Line 2 of the template becomes

   ```
   setwd("C:/Files/Writing/Multimix F & R/Template/CancerT")
   ```

   Note that the backslashes have been replaced by slashes.

2. Line 6 of the template becomes

   ```
   qq <- 2   # Number of clusters
   ```

   As there are two stages of Cancer in the data, we seek a solution with 2 clusters.

3. Line 8 of the template becomes

   ```
   nIt <-  1000    # Maximum number of iterations
   ```

   I tend to use 1000 as a first guess for `nIt`. After the first run it should be clear whether to increase or decrease this.

4. Line 9 of the template becomes

   ```
   eps <- 1e-9   # Minimum increase in loglikelihood per EM step.
   ```

   Setting `eps` too large leads to insufficient iterations, but making it too small can lead to numerical problems.

5. Line 11 of the template becomes

   ```
   cdep  <- NULL # list of MVN cells. [May be NULL.]
   ```

6. Line 12 of the template becomes

   ```
   lcdep <- NULL # list of location cells,
   ```

   We are fitting a local independence model, so both `cdep` and `lcdep` should be set to `NULL`.

7. Nothing to be done here.

8. No problem if extra scripts are loaded. You will soon find out if not enough are loaded.

9. Line 35 of the template becomes

   ```
   rawdf <- read.table("CANCER11a.DAT", header=FALSE)
   ```

   Using `read.table` because the data is not in a `.csv` file, and `header=FALSE` because attribute names are not in the first row of the data file.

10. A suitable definition of `construct` in this case might be

    ```
    construct <- function(rawdf){
    names(rawdf)  = c("Age", "Wt", "PF" , "HX", "SBP", "DBP",  "EKG", "HG",
    "SZ", "SG", "AP", "BM" )
    rawdf[,3] <- as.ordered(rawdf[,3])
    levels(rawdf[,3])  <- c("Active", "Bed49", "Bed51")
    rawdf[,4] <- as.factor(rawdf[,4])
    levels(rawdf[,4])  <- c("No_hist", "Hist")
    rawdf[,7] <- as.factor(rawdf[,7])
    levels(rawdf[,7])  <- c("Normal", "Benign", "Rhythmic", "H_blocks",
    "H_strain", "Old_MCI", "New_MCI")
    rawdf[,12] <- as.factor(rawdf[,12])
    levels(rawdf[,12]) <- c("No_BM", "BM")
    # dframe = transform(rawdf, Age=Age/100, Wt=Wt/100,
    SBP=SBP/10, DBP=DBP/10, HG=HG/100, SG=SG/10)
    dframe = rawdf # scaling does not seem necessary for the cancer df,
    and it makes it harder to interpret parameter values.
    return(dframe)
    }
    ```

    This function names the variables, converts those that are categorical to factors, and names the levels of these factors. If transformations to the continuous variables were thought necessary, either for numerical reasons or to reduce skewness, they might be done here. Actually close inspection of the data file will reveal that some variables have already been transformed.

11. Build starting values for Z matrix and parameter list P.

    In the present example we may use information from `Stage.txt` to give an initial cluster assignment to the patients, and hence calculate a starting `Z` and `P`. The R code might be

    ```
    Stage = scan(file="Stage.txt") # Prostate Cancer Stage 3/4
    Stage = Stage - 2
    Z <- make_Z_discrete(Stage)
    P <- first.Z.to.P(D,Z)    F
    ```

12. Here removing the `"[12.]"` is all you need to do.

13. Here removing the `"[13.]"` is all you need to do.

    The last few lines of the results matrix that are printed give the values at the final iterations of the estimated class probabilities and the log-likelihoods. If the log-likelihoods still appear to be increasing you may wish to re-run with a larger value of `nIt`.

7.2.3. The resulting script to fit the local independence model

After editing the template as described in 7.2.2 you should reach something like the following R script.

```
# Use setwd(folder) to change to a folder in which the Multimix scripts
    are.
setwd("C:/Files/Writing/Multimix F & R/Template/CancerT")

# Global constants which might be altered between different runs on the
    same data.

qq <-   2   # Number of clusters [using "q" would conflict with R's quit
    function]

nIt <-  1000  # Maximum number of iterations
eps <-  1e-9  # Minimum increase in loglikelihood per EM step. [stopping
    criterion]

cdep <- NULL   # list of MVN cells. [may be NULL]
lcdep <- NULL  # list of location cells.
# discrete first in each cell. Also may be NULL.
# Note: Once the data frame has been set up, check
# that the values of cdep and lcdep model the
# correct variable associations for the model you
# are fitting.

library(mvtnorm)

# Load Multimix scripts (function definitions)
```

```
# These scripts should not be data-dependent.

source("Functions.r")
source("data_organise.r")
source("make_Z_discrete.r")
source("make_Z_random.r")
source("first.Z.to.P.r")
source("P.to.Z.r")
source("Z.to.P.r")
source("mmain2.r")

# Read data csv or txt into raw dataframe

rawdf <-read.table("CANCER11a.DAT", header=FALSE)

# Make function converting raw dataframe into Multimix-ready dataframe.
# Categorical variables should be changed to factors with appropriate
    labels.
# Continuous variables may need to be scaled or transformed.

construct <- function(rawdf){
names(rawdf)  = c("Age", "Wt", "PF" , "HX", "SBP", "DBP",  "EKG", "HG", "
    SZ", "SG", "AP", "BM" )
rawdf[,3] <- as.ordered(rawdf[,3])
levels(rawdf[,3])  <- c("Active", "Bed49", "Bed51")
rawdf[,4] <- as.factor(rawdf[,4])
levels(rawdf[,4])  <- c("No_hist", "Hist")
rawdf[,7] <- as.factor(rawdf[,7])
levels(rawdf[,7])  <- c("Normal", "Benign", "Rhythmic", "H_blocks", "H_
    strain", "Old_MCI", "New_MCI")
rawdf[,12] <- as.factor(rawdf[,12])
levels(rawdf[,12]) <- c("No_BM", "BM")
# dframe = transform(rawdf, Age=Age/100, Wt=Wt/100, SBP=SBP/10, DBP=DBP/
    10, HG=HG/100, SG=SG/10)
dframe = rawdf # scaling does not seem necessary for the cancer df, and
    it makes it harder to interpret parameter values.
return(dframe)
}

# Apply function to raw dataframe.

dframe <- construct(rawdf)

# Save dimensions of 'dframe'.

n <- dim(dframe)[1]
v <- dim(dframe)[2]

# Build list of needed functions of the data

D <- data_organise(dframe, cdep, lcdep, minpstar = 1e-9)

# Build starting values for Z matrix and parameter list P.

```

```
      #(Use "make_Z_random.r" or similar script to construct a starting Z
    matrix, then use "first.Z.to.P.r" to construct P.)

Stage = scan(file="Stage.txt") # Prostate Cancer Stage 3/4
Stage = Stage - 2
Z <- make_Z_discrete(Stage)
P <- first.Z.to.P(D,Z)

      #(Alternatively, if starting from a parameter list P, use the
   script "P.to.Z.r" to construct the initial Z.)

# Now starting values of Z and P should have been constructed.

#str(Z)
#P[-16]

# Apply mmain() function

zpr <- mmain(D,Z,P)

# Trim results matrix.

ok <- apply(zpr$results[,1:qq + 1],1,sum) > 0.5
zpr$results <- zpr$results[ok,]
tail(zpr$results)
```

7.2.4. Discussion of results

Running the R script in 7.2.3 creates an output list `zpr` with 3 members

```
zpr$Z
zpr$P
zpr$results
```

and the last few rows of the third member are printed:

```
> tail(zpr$results)
           [,1]      [,2]      [,3] [,4]
[24,] -11380.68 0.5636371 0.4363629   24
[25,] -11380.68 0.5636358 0.4363642   25
[26,] -11380.68 0.5636348 0.4363652   26
[27,] -11380.68 0.5636342 0.4363658   27
[28,] -11380.68 0.5636337 0.4363663   28
[29,] -11380.68 0.5636334 0.4363666   29
```

This shows us that 29 iterations were carried out. The first column gives the log-likelihood values at iterations 24 to 29. These are constant to 2 decimal places. The second and third columns give the estimated probability of a patient being in cluster 1 or 2 respectively. By iteration 29 these probabilities have settled down to about 5 decimal places.

If we wished for more accuracy we could re-run the script taking `eps` to be a smaller positive value. However let us suppose that we are satisfied with these estimates.

The parameters for the discrete variables (`PF,HX,EKG,BM`, Patient Activity,Family History of Cancer,Electrocardiogram code,Bone Metastases) are given as follows:

```
> zpr$P$dstat
[[1]]
     dframe[, dlink[i]]Active dframe[, dlink[i]]Bed49 dframe[, dlink[i]]Bed51
[1,]                0.9403458             0.05228008             0.007374072
[2,]                0.8502996             0.08685727             0.062843172

[[2]]
     dframe[, dlink[i]]No_hist dframe[, dlink[i]]Hist
[1,]                 0.4930971              0.5069029
[2,]                 0.6560642              0.3439358

[[3]]
     dframe[, dlink[i]]Normal dframe[, dlink[i]]Benign
[1,]                0.3321646               0.04587239
[2,]                0.3477083               0.05171303
     dframe[, dlink[i]]Rhythmic dframe[, dlink[i]]H_blocks
[1,]                  0.0752608                 0.06069727
[2,]                  0.1440157                 0.04221353
     dframe[, dlink[i]]H_strain dframe[, dlink[i]]Old_MCI
[1,]                  0.3111946                 0.1710751
[2,]                  0.2976018                 0.1167477
     dframe[, dlink[i]]New_MCI
[1,]               0.003735166
[2,]               0.000000000

[[4]]
     dframe[, dlink[i]]No_BM dframe[, dlink[i]]BM
[1,]               0.9920376          0.007962357
[2,]               0.6387961          0.361203928
```

We can interpret these results as follows:

Patient Activity Patients in Cluster 1 tend to be more active than those in Cluster 2, spending less of the day in bed.

History Patients in Cluster 1 are more likely than those in Cluster 2 to have a Family History of Cancer.

Electrocardiogram code While many proportions are similar in the two clusters, Cluster 2 has a higher proportion of `Rhythmic` codes and Cluster 1 has a higher proportion of `Old_MCI` codes.

Bone Metastases Bone metastases are rare in Cluster 1, but very common in Cluster 2.

The parameters for the continuous variables are

```
> zpr$P$ostat
          Age         Wt       SBP       DBP        HG        SZ         SG        AP
[1,] 71.60951 100.45473 14.54641 8.293449 138.1190 2.896396  8.928559 1.646654
[2,] 71.48641  97.15477 14.15781 7.982806 129.1332 4.140390 12.073834 3.921064
> sqrt(zpr$P$ovar)
          Age       Wt       SBP       DBP        HG       SZ       SG        AP
[1,] 6.657555 13.07395 2.557355 1.543339 17.81287 1.201406 1.155319 0.5002082
[2,] 7.228751 13.41991 2.232384 1.336409 20.09283 1.701843 1.424942 1.6974494
```

Age Little difference in mean age between clusters.

Weight Higher mean weight in Cluster 1.

Systolic BP Higher mean SBP in Cluster 1.

Diastolic BP Higher mean DBP in Cluster 1.

Haemoglobin Higher mean HG in Cluster 1.

Tumour Size Higher mean SZ in Cluster 2.

Tumour Index Higher mean SG in Cluster 2.

Acid Phos. Higher mean AP in Cluster 2.

The object `zpr$` contains information about the assignment of observations to cluster, specifically the matrix `Z`.

```
> head(zpr$Z)
          [,1]         [,2]
[1,] 0.9992327 0.0007672614
[2,] 0.9986436 0.0013564100
[3,] 0.9986733 0.0013266619
[4,] 0.9946733 0.0053266981
[5,] 0.9687294 0.0312705560
[6,] 0.9929612 0.0070388029
> plot(zpr$Z[,1])
```

Figure 7.1 contains a plot of the first column of `Z` in the local independence model. To produce this plot from R use the command `plot(zpr$Z[,1])`.

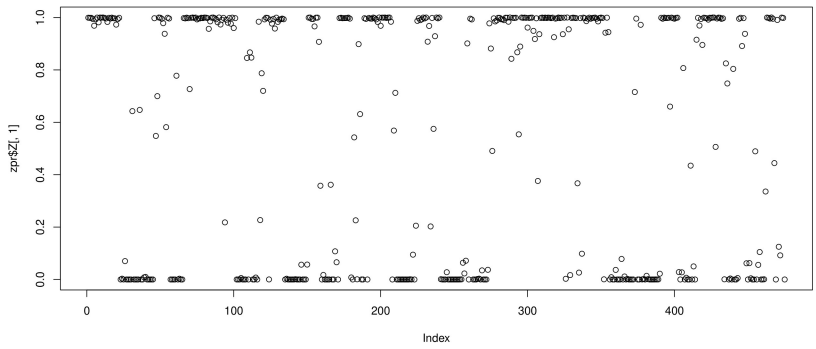

Figure 7.1.: Probability of membership in Cluster 1

7.2.5. Fitting a model with a local dependency to the Prostate Cancer data

It may well be doubted that the two blood pressure variables are independent, even locally by cluster, bearing in mind the almost mechanical relationship between them. This doubt is supported if we assign patients to clusters 1 and 2 by which column of `Z` has the higher value and calculate the correlation coefficient between `SBP` and `DBP` within each cluster. The values are 0.63 and 0.62 in clusters 1 and 2 respectively, based on 272 and 203 observations.

We now fit a new model in which these within-cluster dependencies are incorporated. Only one line of R code needs to be changed. The value of `cdep` needs to be changed. The new line of code is

```
cdep <- list(c(5,6))
```

which causes the fitting of a bivariate normal distribution to the pair `SBP,DBP` within each cluster. (Note that the variables `SBP,DBP` are in columns 5 and 6 of `dframe`.)

This time the algorithm converges in 28 iterations, printing

```
> tail(zpr$results)
           [,1]      [,2]      [,3] [,4]
[23,] -11263.14 0.5640084 0.4359916   23
[24,] -11263.14 0.5640070 0.4359930   24
[25,] -11263.14 0.5640059 0.4359941   25
[26,] -11263.14 0.5640052 0.4359948   26
[27,] -11263.14 0.5640048 0.4359952   27
[28,] -11263.14 0.5640044 0.4359956   28
```

Supposing that we are unhappy with the convergence of the cluster proportions (Columns 2 and 3 in the output), let's run this again with `eps` set to `1e-10`.

```
> tail(zpr$results)
           [,1]      [,2]      [,3] [,4]
[26,] -11263.14 0.5640052 0.4359948   26
[27,] -11263.14 0.5640048 0.4359952   27
[28,] -11263.14 0.5640044 0.4359956   28
[29,] -11263.14 0.5640042 0.4359958   29
[30,] -11263.14 0.5640040 0.4359960   30
[31,] -11263.14 0.5640039 0.4359961   31
```

This time the mean and variance parameters for `SBP` and `DBP` do not appear in `zpr$P$ostat or \verbzpr`P`ovar|`, but as follows

```
> zpr$P$cstat
[[1]]
          SBP      DBP
[1,] 14.53338 8.285092
[2,] 14.17434 7.993353

> zpr$P$cvar
[[1]]
          SBP      DBP
[1,] 6.517203 2.374890
[2,] 5.024269 1.800977

> zpr$P$MVMV
[[1]]
[[1]][[1]]
         [,1]     [,2]
[1,] 6.517203 2.495988
[2,] 2.495988 2.374890

[[1]][[2]]
         [,1]     [,2]
[1,] 5.024269 1.843676
[2,] 1.843676 1.800977
```

Note how the variance parameters reappear as the diagonal members of the variance/-covariance matrices for each cluster.

The other model parameters are not printed here. Numerically they are similar to the parameters for the local independence model and are printed by the same commands.

Figure 7.2 contains a plot of the first column of `Z` in the BP model. To produce this plot from R use the command `plot(zpr$Z[,1])`.

Comparing the log-likelihoods of the local independence (LI) and the blood pressure (BP) models we can calculate the *Deviance* as

$$2(\lambda_{BP} - \lambda_{LI}) = 2(-11263.14 + 11380.68) = 235.08$$

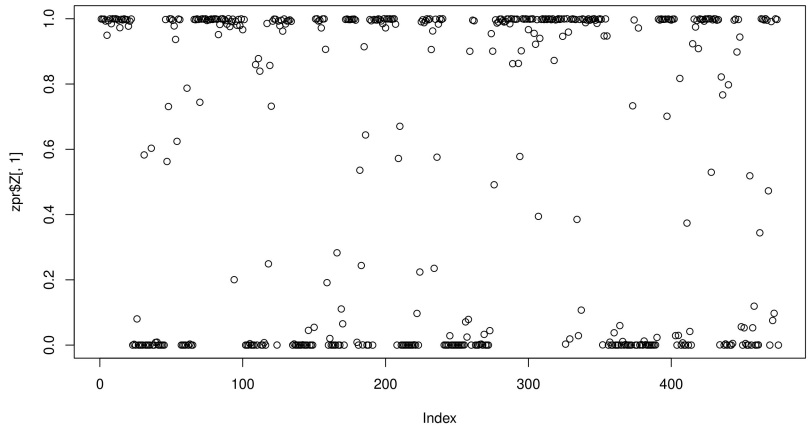

Figure 7.2.: Probability of membership in Cluster 1 for BP model

As the two models are nested and BP has two more parameters than LI, we might hope that the Deviance under the LI model would be χ^2_2 distributed. However this is not true for mixture models. Just the same, the apparently large Deviance is evidence for the BP model.

7.2.6. Fitting a model including Olkin-Tate dependencies to the Prostate Cancer Data

We will now modify the R script of Subsection 7.2.5 by changing the value of `lcdep` by means of

```
lcdep <- list(c(12, 2, 8), c(3,1))
```

The variables 12, 2, and 8 are Bone Metastases, Weight, and Haemoglobin; the variables 3 and 1 are Performance and Age. In both groups the discrete variable is listed first. These two groups were chosen to be Olkin-Tate distributed as

- Neither group overlaps with the Blood Pressure group;
- Nor with each other;
- Each group shows moderately large within-cluster correlations.

The fitting of the OT model terminates after 28 iteration steps giving

```
> tail(zpr$results)
           [,1]      [,2]      [,3] [,4]
[23,] -11227.18 0.5563988 0.4436012   23
```

```
[24,] -11227.18 0.5563983 0.4436017   24
[25,] -11227.18 0.5563979 0.4436021   25
[26,] -11227.18 0.5563977 0.4436023   26
[27,] -11227.18 0.5563975 0.4436025   27
[28,] -11227.18 0.5563974 0.4436026   28.
```

The proportion in Cluster 2 is slightly higher than in the previous models.

Again we find the parameter estimates for the discrete variable distributions in `zpr$P$dstat`.

```
> zpr$P$dstat
[[1]]
     dframe[, dlink[i]]No_hist dframe[, dlink[i]]Hist
[1,]                 0.4921096              0.5078904
[2,]                 0.6546445              0.3453555

[[2]]
     dframe[, dlink[i]]Normal dframe[, dlink[i]]Benign
[1,]                0.3290931              0.04670664
[2,]                0.3513073              0.05057138
     dframe[, dlink[i]]Rhythmic dframe[, dlink[i]]H_blocks
[1,]                 0.07590462                 0.06097878
[2,]                 0.14208668                 0.04216194
     dframe[, dlink[i]]H_strain dframe[, dlink[i]]Old_MCI
[1,]                  0.3120814                 0.1714517
[2,]                  0.2967112                 0.1171615
     dframe[, dlink[i]]New_MCI
[1,]               0.003783741
[2,]               0.000000000
```

Note that the parameters for Bone Metastases and Performance are not printed here, they will be included with those for the Olkin-Tate distributions.

Now the parameters for the univariate normals:

```
> zpr$P$ostat
          SZ        SG       AP
[1,] 2.895325  8.901892 1.635087
[2,] 4.121443 12.055977 3.898473
> zpr$P$ovar
          SZ       SG        AP
[1,] 1.429882 1.288211 0.2379408
[2,] 2.911036 2.027254 2.8715067,
```

and the bivariate normal:

```
> zpr$P$cstat
[[1]]
```

```
          SBP      DBP
[1,] 14.54408 8.286649      means
[2,] 14.16708 7.996402

> zpr$P$MVMV
[[1]]
[[1]][[1]]
         [,1]     [,2]
[1,] 6.573721 2.512831      cluster 1 covariance matrix
[2,] 2.512831 2.387751

[[1]][[2]]
         [,1]     [,2]
[1,] 4.971357 1.830917      cluster 2 covariance matrix
[2,] 1.830917 1.794995.
```

Now discrete Olkin-Tate parameters for (1) Bone Metastases and (2) Performance:

```
> zpr$P$ldstat
[[1]]
     dframe[, lcdisc[i]]No_BM dframe[, lcdisc[i]]BM
[1,]                0.9901237           0.009876322
[2,]                0.6469586           0.353041381

[[2]]
     dframe[, lcdisc[i]]Active dframe[, lcdisc[i]]Bed49
[1,]                 0.9427328               0.04983234
[2,]                 0.8487744               0.08936339
     dframe[, lcdisc[i]]Bed51
[1,]              0.007434823
[2,]              0.061862187.
```

Now continuous Olkin-Tate parameters(means) for (1) Bone Metastases and (2) Performance:

```
> zpr$P$lcstat
[[1]]
[[1]][[1]]
           Wt       HG
[1,] 100.0456 137.7718          No_BM
[2,] 100.1618 134.8070

[[1]][[2]]
            Wt       HG
[1,] 114.55631 145.1584         BM
```

```
[2,]  92.74101 120.1254


[[2]]
[[2]][[1]]
         [,1]
[1,] 71.43329           Active
[2,] 71.49788

[[2]][[2]]
         [,1]
[1,] 76.17371           Bed49
[2,] 69.39944

[[2]][[3]]
         [,1]
[1,] 74.97245           Bed51
[2,] 72.62595
```

Now the covariance matrices for the two Olkin-Tate distributions:

```
> zpr$P$LMV
[[1]]
[[1]][[1]]
         [,1]     [,2]
[1,] 164.6210  46.0469          cluster 1 covariance matrix
[2,]  46.0469 324.6208          Wt by HG

[[1]][[2]]
          [,1]      [,2]
[1,] 174.83410  55.85055        cluster 2 covariance matrix
[2,]  55.85055 350.39225        Wt by HG


[[2]]
[[2]][[1]]
         [,1]
[1,] 42.42664                   cluster 1 variance (Age)

[[2]][[2]]
         [,1]
[1,] 52.56374                   cluster 2 variance (Age)
```

Figure 7.3 contains a plot of the first column of `Z` in the OT model. To produce this plot from R use the command `plot(zpr$Z[,1])`.

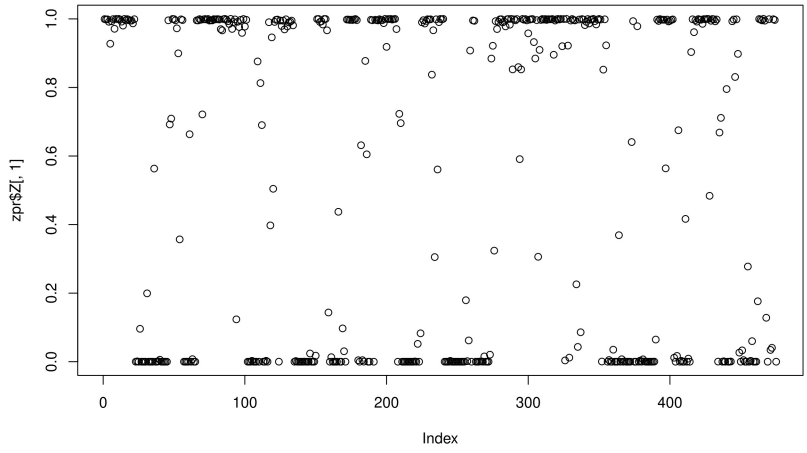

Figure 7.3.: Probability of membership in Cluster 1 for the OT model

7.3. Fortran Software for fitting Multimix

This may be used by both people who are familiar with the Fortran language and also by those with minimal knowledge of it. Naturally the former group may be able to do more things with the Fortran program, by quite a lot may be done by the latter group.

There may be some speed advantage in using the Fortran version when fitting complex models to larger data sets but the R version is fast enough for the examples in this guide. The authors would be very interested in examples where the two versions differ greatly in execution speed. If you find such an example we would be very interested in learning about the data and the model that you were fitting.

There are some advantages to knowing both versions of `Multimix` and we shall show how to switch from one version to another simply.

7.3.1. The gfortran compiler

This guide assumes that the **gfortran** compiler has been installed on a Windows computer. It should not be difficult tor the reader to modify our examples to other Fortran compilers or other platforms.

The **gfortran** compiler is part of the GNU Compiler Collection (GCC), see *https://gcc.gnu.org/wiki/GFortran* for more information.

For information about installing **gfortran** on a range of computers see `https://fortran-lang.org/en/learn/os_setup/install_gfortran/`.

You may already have **gfortran** on your Windows computer. To check this, open a PowerShell window and type `cmd` and press Enter.

If **gfortran** is installed you should get the response:

```
gfortran: fatal error: no input files
compilation terminated.
```

Otherwise you may need to install **gfortran**.

If you have a Windows computer you should consider installing an appropriate version of Rtools on your computer. See `https://cran.r-project.org/bin/windows/Rtools/rtools42/rtools.html`.

Rtools includes a number of programming tools for developers including Mingw-w64 (`https://en.wikipedia.org/wiki/Mingw-w64`). Mingw-w64 includes a number of GCC compilers, including gfortran.

7.4. A Fortran program for fitting Multimix

Before the Fortran version of `Multimix` can be used you must create a working directory. Here we will assume that the working directory is `C:/Files/Writing/Multimix F & R/Template/CancerT`.

Some conditions apply to the working directory:

Source File The Fortran source file, here called *multimix90.f90*, should be present.

Executable File This file, often called *a.exe*, will be created when the source file is compiled by gfortran.

Data File This is an $n \times p$ fully numeric file containing the data.

Parameter File This is a file whose purpose is to describe the particular `Multimix` model being fitted. It is a fully numeric text file. The format of this file is described in section [].

Output Files These files, called `GENERAL.OUT` and `GROUPS.OUT`, contain the `Multimix` output after the executable file has been run. They should **NOT** be present before the executable has been run. The format of these files is described in section [].

After you have set up your working directory, you can run `Multimix` using Fortran in a similar way to this:

```
PS C:\Files\Writing\Multimix\nov17-bpLoc> cmd
Microsoft Windows [Version 10.0.19043.2006]
(c) Microsoft Corporation. All rights reserved.

C:\Files\Writing\Multimix\nov17-bpLoc>gfortran multimix90.f90

C:\Files\Writing\Multimix\nov17-bpLoc>a.exe
  MIXTURE ESTIMATION BY EM
 ------------------------------
 Data file:
```