Analysis

Team Student

This package is part of a project to determine the best way to evaluate data with rounded values (binned data). Our task was to use an EM algorithm with penalty for mixtures of t-distributions.

This vignette has the goal to explain all important functions contained in this package and their usage as well as the visualization with examples.

```
library(t.mix)
```

Data

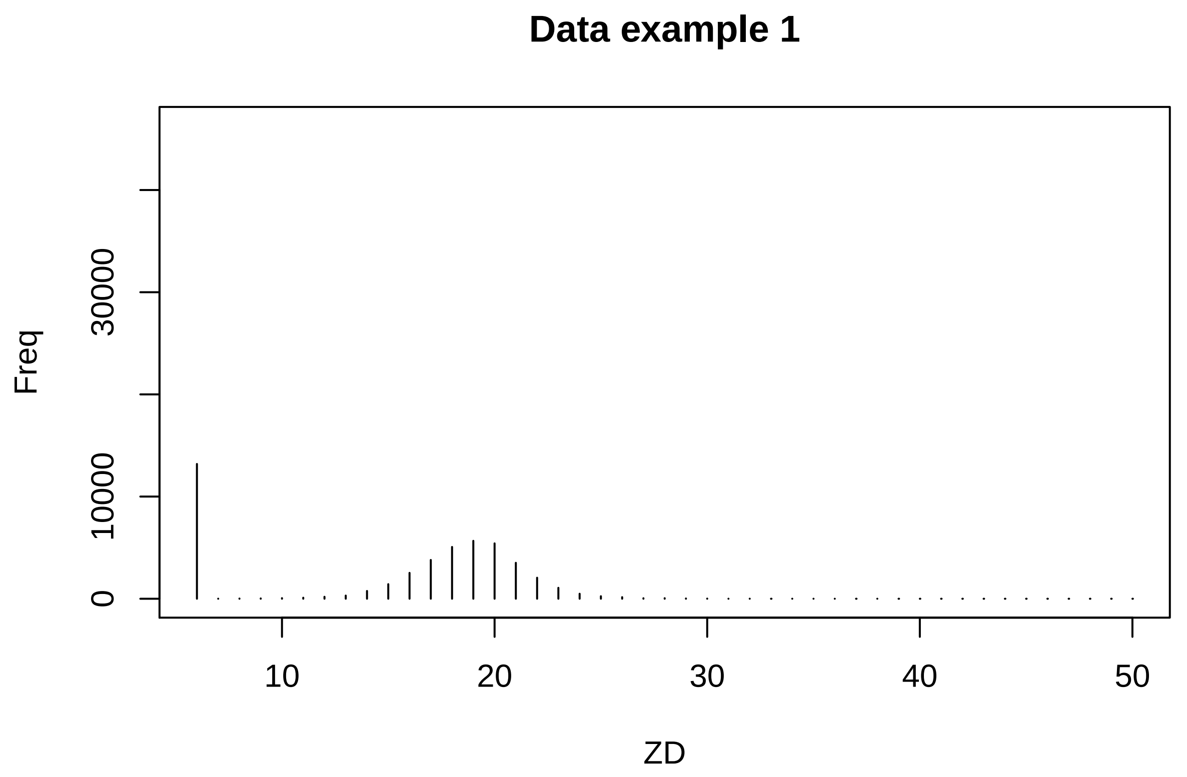
```
ZD <- read.csv2("../ZD.csv")
#To check if it read in correctly:
head(ZD[1])
#>   Antimicrobial
#> 1      Amikacin
#> 2      Amikacin
#> 3      Amikacin
#> 4      Amikacin
#> 5      Amikacin
#> 6      Amikacin
```

First we establish three example data sets, which will be used throughout this vignette. These datasets are subsets of our original data.

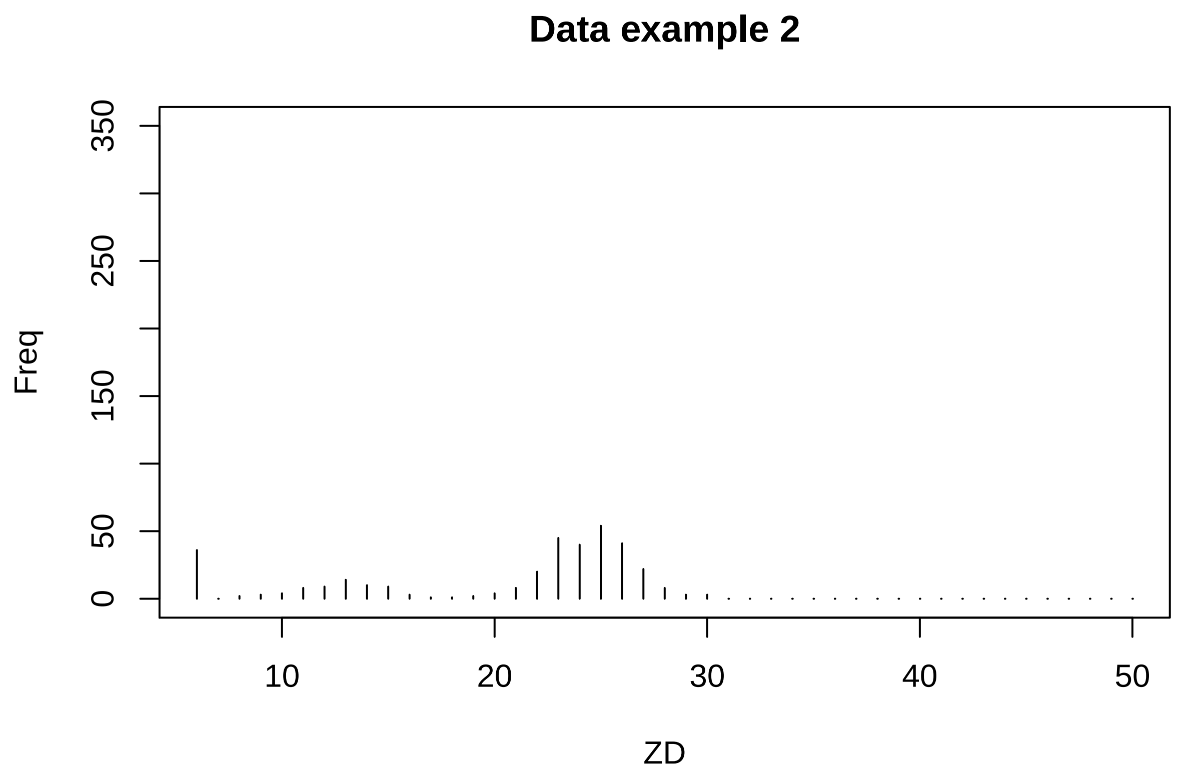
```
ZDs1 <- subset(ZD, Antimicrobial == "Ampicillin" & Bacterium == "Escherichia coli")

example1 <- data.frame(ZD = as.integer(gsub("^Z", "", colnames(ZDs1))),
                       Freq = unname(unlist(ZDs1)))
#> Warning in data.frame(ZD = as.integer(gsub("^Z", "", colnames(ZDs1))), Freq
#> = unname(unlist(ZDs1))): NAs durch Umwandlung erzeugt

ZDs2 <- subset(ZD, Antimicrobial == "Piperacillin" & Bacterium == "Escherichia coli")
example2 <- data.frame(ZD = as.integer(gsub("^Z", "", colnames(ZDs2))),
                       Freq = unname(unlist(ZDs2)))
#> Warning in data.frame(ZD = as.integer(gsub("^Z", "", colnames(ZDs2))), Freq
#> = unname(unlist(ZDs2))): NAs durch Umwandlung erzeugt

ZDs3 <- subset(ZD, Antimicrobial == "Mecillinam" & Bacterium == "Escherichia coli")

example3 <- data.frame(ZD = as.integer(gsub("^Z", "", colnames(ZDs3))),
                       Freq = unname(unlist(ZDs3)))
#> Warning in data.frame(ZD = as.integer(gsub("^Z", "", colnames(ZDs3))), Freq
#> = unname(unlist(ZDs3))): NAs durch Umwandlung erzeugt
```

The following plots ought to give an insight to the example-data (45 obsvervations of rounded values and their respective frequencies) used here.

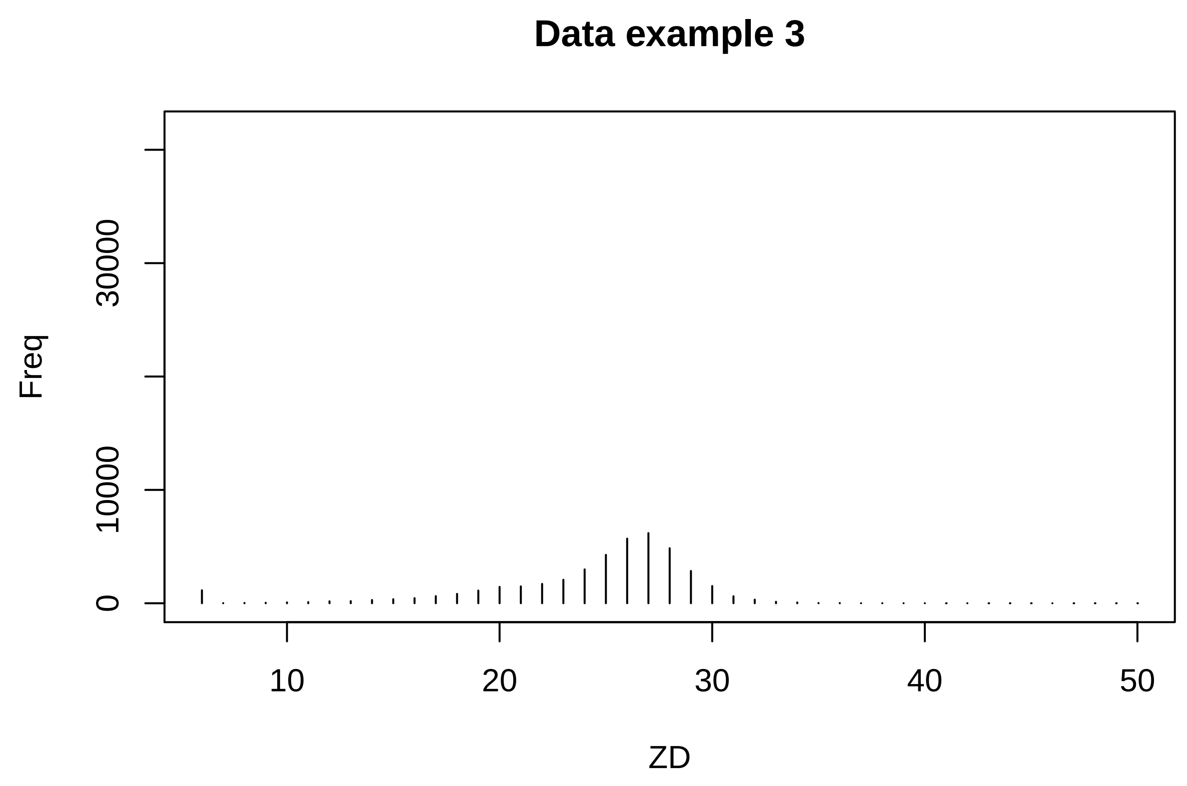
```
plot(Freq ~ ZD, data = example1, type = "h", main = "Data example 1")
```



```r
plot(Freq ~ ZD, data = example2, type = "h", main = "Data example 2")
```



```r
plot(Freq ~ ZD, data = example3, type = "h", main = "Data example 3")
```



Distribution Functions

Let's start with a few complementary functions: In case we want to test our functions we use testdata. In our case we build our own functions to so, namely: **rt.mixture, dt.mixture, pt.mixture and qt.mixture**.

rt.mixture simulates N observations from a mixture of t distributions. It gives a numeric vector of length N.

```
pi=c(0.6, 0.4)
mu=c(10, 30)
s=c(4, 7)
df=c(4, 300)
```

The parameters π, μ, σ, ν (here "df") do not have to be a single numeric value. They can also be vectors of the same length. Their values must be *positive* and the sum of the values of pi has to be equal to 1! There are no default-values for these parameters!

```
test <- rt.mixture(N=1000, pi = pi, mu = mu, s = s, df = df, round=TRUE)
summary(test)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  -15.00    9.00   14.00   18.04   28.00   48.00
```

dt.mixture gives the PDF of a mixture of t distributions. The parameter y must be a numeric vector at which the PDF shall be evaluated.

```
test2 <- dt.mixture(test, pi, mu, s, df)
```

pt.mixture gives the CDF of a mixture of t distributions. Similiar to **dt.mixture** the parameter y is a numeric vector at which the CDF shall be evaluated.

```
test3 <- pt.mixture(test, pi, mu, s, df)
```

Lastly **qt.mixture** is the quantile function of a mixture of t distributions. The parameter p is the value at which the quantile function shall be evaluated. It must be a numeric scalar within $(0, 1)$. *tol* is the numerical tolerance of the result. It also must be a small, positive, numeric scalar. In contrast to the other parameters it does have a default value.

```
qt.mixture(0.8, pi, mu, s, df)
#> [1] 10
#> [1] 19.78603
#> [1] 30.09698
```

Main functions for analysis

To get an idea of how our functions work, we provided an overview of our algorithm, which should help to understand the functions and their purpose.

Algorithm 1 EM Algorithm for mixtures of t's with binned data (grouped)

```
1: procedure
2:     Calculate starting values $\pi_k^{(0)}, \mu_k^{(0)}, \sigma_k^{(0)}, \nu_k^{(0)}, \quad k = 1, ..., K$
3:     Set $\alpha_0$, $\beta_0$
4:     repeat
5:         $\tau_{k,j1}^{(i)} = \frac{\pi_k^{(i)} f_{\tilde{Y}}(j|\mu_k^{(i)},\sigma_k^{(i)},\nu_k^{(i)})}{\sum_{c=1}^{K} \pi_c^{(i)} f_{\tilde{Y}}(j|\mu_c^{(i)},\sigma_c^{(i)},\nu_c^{(i)})}$ for $k = 1, ..., K,\ j = 1, ..., J$
6:         $\pi_k^{(i+1)} \leftarrow \frac{1}{\tilde{N}} \sum_{j=1}^{J} n_j \tau_{k,j1}^{(i)} \qquad$ for $k = 1, ..., K$
7:         $(\mu_k^{(i+1)}, \sigma_k^{(i+1)}, \nu_k^{(i+1)}) \leftarrow \arg\max Q_2(\mu,\ \sigma,\ \nu;\ \Theta^{(i)}) +$
   $Q_3(\sigma;\ \alpha_0, \beta_0)$ for $k = 1, ..., K$
8:     until convergence criterion is fulfilled
9:     return $\Theta^{(i+1)}$
```

The two main functions to analyse the data are **total.function** and **model.select.t.mixture**. Most of the other functions included in this package do the preliminary work or are used in either one or the other or both of the main functions.

We will now proceed with evaluating the first example dataset.

```
#The observed values are the same for all 3 examples:
j <- c(6:50)
j
#>  [1]  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
#> [24] 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
#The n, of course, vary and have to be defined individually for each example.
n1 <- as.integer(ZDs1[4:48])
n1
#>  [1] 13191     6    20    30    60   102   191   311   761  1426  2533
#> [12]  3796  5070  5667  5420  3518  2058  1069   489   248   155    49
#> [23]    54    24    14     4     6     0     4     2     1     0     2
#> [34]     0     0     0     0     0     0     0     0     0     0     0
#> [45]     0
```

After defining ‘n’ and ‘j’, we now have to find out the best number for the components.

```
#With one component:
(result1 = total.function(n1, j, K=1, atoms=6, memb.exp=2, draw=TRUE))
#Two components:
(result1 = total.function(n1, j, K=2, atoms=6, memb.exp=2, draw=TRUE))
#Three components:
(result1 = total.function(n1, j, K=3, atoms=6, memb.exp=2, draw=TRUE))
```

The last step is the model selection process:

```
#model selection (up to Kmax=5)
(selected1 <- model.select.t.mixture(n1, j, Kmax=5, memb.exp=2, atoms=6))
```

This process is repeated for example dataset 2:

```
####### Second data set ##########

n2 = as.integer(ZDs2[4:48])

#one component
(result2 = total.function(n2, j, K=1, atoms=6, memb.exp=2, draw=TRUE))

#two components
(result2 = total.function(n2, j, K=2, atoms=6, memb.exp=2, draw=TRUE))

#three components
(result2 = total.function(n2, j, K=5, atoms=6, memb.exp=2, draw=TRUE))

#model selection (up to Kmax=5)
(selected2 <- model.select.t.mixture(n2, j, 5, memb.exp=2, atoms=6))
```

On to example 3.

```
n3 = as.integer(ZDs3[4:48])

#one component
(result3 = total.function(n3, j, K=1, atoms=6, memb.exp=2, draw=TRUE))

#two components
(result3 = total.function(n3, j, K=2, atoms=6, memb.exp=2, draw=TRUE))

#three components
(result3 = total.function(n3, j, K=3, atoms=6, memb.exp=2, draw=TRUE))

#four components
(result3 = total.function(n3, j, K=4, atoms=6, memb.exp=2, draw=TRUE))

#model selection (up to Kmax=5)
(selected3 <- model.select.t.mixture(n3, j, Kmax=5, memb.exp =2, atoms=6, optim.method="Nelder-Mead"))
```