trajeR, an **R** package for cluster analysis of time series

Cédric NOEL & JANG SCHILTZ

University of Luxembourg
Department of Finance
University of Luxembourg
6, rue Richard Coudenhove-Kalergi
L-1359 Luxembourg, Luxembourg
E-mail: jang.schiltz@uni.lu

Abstract

We present the **R** package **trajeR** which provides all necessary tools to calibrate generalized finite mixture models, plot the results graphically and test the model adequacy. First, we give an overview of the generalized finite mixture model for clustering time series and describe the core function `trajeR` of the package **trajeR**. We give a step-by-step guideline on how to use the different available functions on simulated data sets with the three main underlying data distributions that appear in practice, the binary logit distribution, the (censored) normal distribution and the zero inflated Poisson distribution. Even though the estimation results are usually very similar, the **trajeR** package supports the two main estimations methods which are direct optimization of the derivative of the loglikelihood and the Expected Maximization Algorithm. It also allows to compute group membership probabilities for all subjects in the data set, to exhibit profiles for the clusters found in the analysis and provides a whole bunch of model adequacy criteria, together with the main model selection criteria.

Keywords: **R** package, generalized finite mixture model, fuzzy logic cluster analysis of trajectories

1 Introduction

Longitudinal data are the starting point for empirical research on numerous subjects in finance, economics, sociology, psychology, criminology and medicine and a host of statistical techniques are available for analyzing them (see Singer & Willet 2003, Gibbons, Hedeker & DuToit 2010 or Locascio & Atri 2011 for the general case or Verbeken, Fieuws & Molenberghs 2014 for multivariate data). The common statistical aim of these various application fields is the modelization of the evolution of an age or time-based phenomenon (Nagin 2002). Hence, the study of developmental trajectories is a central theme (Jones, Nagin & Roeder 2001, Sampson & Laub 2005). The objective of these approaches is to capture information about interindividual differences in intraindividual change over time (Nesselroade 1991). In the 1990s, the generalized mixed model assuming a normal distribution of unobserved heterogeneity (Bryk & Raudenbush 1992), multilevel modeling (Goldstein 1995), latent growth curves modeling (Muthén 1989, Willett & Sayer 1994) and the nonparametric mixture model, based on a discrete distribution of heterogeneity (Jones, Nagin & Roeder 2001) have emerged.

Growth mixture modeling, introduced by Muthen & Shedden (1999), is a very suitable framework to handle the issue of unobserved heterogeneity. Latent class growth analysis, also called nonparametric mixed model or semi-parametric mixture model is the simplest specification of a growth mixture model. It allows no variation across individuals within classes. It was originally discussed by Nagin & Land (1993), Nagin (1999) and Roeder, Lynch & Nagin (1999) and is actually specifically designed to detect the presence of distinct subgroups among a set of trajectories and represents an interesting compromise between analysis around a single mean trajectory and case studies (VonEye & Bergman). Compared to subjective classification methods, the nonparametric mixed model has the advantage of providing a formal framework for testing the existence of distinct groups of trajectories. Andruff et ali (2009) conclude that latent class growth analysis serves as a steppingstone to growth mixture modeling analyzes in which the precise number and shape of each trajectory must be known a priori in order for the researcher to impute the requisite start values for the model to converge in software packages such as **Mplus** (Jung & Wickrama 2008).

While the conceptual aim of the analysis is to identify clusters of individuals with similar trajectories, the model's estimated parameters are not the result of a classical cluster analysis but of maximum likelihood estimation (Nagin 2005). Moreover, this method allows to evaluate the accuracy of the assignment of the individuals to the different sub-groups and to consider the variation of this accuracy in subsequent analyses (Dupéré et ali 2007). Nagin & Odgers (2010) document numerous applications of group-based trajectory modeling in criminology and clinical research. They state that the appeal of group-based trajectory modeling for the future lies in the potential for the innovative application of trajectory models on their own, in conjunction with other statistical methods or embedded within creative study designs while carefully considering the perils and pitfalls inherent in the use of any methodology. Let's also point out these finite mixture models, which are actually fuzzy logic cluster analysis of time series models should not be confounded with finite mixture models in the sense of McLachlan & Peel (2000), which refer to mixtures of probability distributions.

Schiltz (2015) presents a generalization of Nagin's finite mixture model that allows non parallel trajectories for different values of covariates and different standard deviations for the disturbances in the different groups.

This paper presents an **R** package (R Core Team 2017) that analyzes longitudinal data by fitting the general finite mixture model. It can be seen as an extension of the **SAS** (Delwiche & Slaughter: 2019) procedure **Proc Traj** developed by Jones, Nagin & Roeder (2001), Jones & Nagin (2007) and their **Stata** (Acock 2018) version (Jones & Nagin 2012) to the generalized finite mixture model, with the additional advantage that, as an **R** package, it is open source software and can be adapted to the needs of the users.

The remainder of this article is structured as follows. In section two, we present the generalized finite mixture model and in section three the design of our **R** package. Sections four to six are dedicated to show in detail how the package can be used for the three main distributions underlying the data in this model. More precisely, in section four, we present the Logit model, in section five the censored normal model and in section six the zero inflated Poisson model. Section seven is dedicated to the possibility of modeling nonlinear trajectories for the normal model. Section eight finally presents some other important aspects of the generalized finite mixture model that can be treated with our **R** package, such as group membership probabilities, group profiles, model selection criteria and model adequacy criteria. A summery with an outlook on future extensions then concludes this paper in section nine.

2 The generalized finite mixture model

We consider a population of size N divided into K latent classes, the assignment of an individual into which is based on the degree of similarity of the developmental trajectories of the individuals. This models supposes no structural between-subject variability within a class, hence the error variance is assumed to be constant inside a given class.

More precisely, consider a time-varying variable of interest Y defined in a population Ω of size N. Let $Y_i = y_{i_1}, \cdots, y_{i_T}$ be T measures of the variable Y, taken at times $t_1, \cdots, t_T$ for subject number i belonging to a sample of size n.

The aim of the analysis is to divide the population into K sub-populations $G_1, ...G_K$, which are homogeneous in the sense that two subjects in the same group have similar trajectories for the variable of interest Y and two subjects in different groups have quite different trajectories for the variable of interest Y.

Let $P^k(Y_i)$ be the probability of Y_i given membership in group G_k and $P(Y_i)$ the unconditional probability of observing the realization Y_i of Y. Furthermore, for a given group G_k, we suppose conditional independence for the sequential realizations of the elements y_{i_t} over the T periods of measurements. Then,

$$P(Y_i) = \sum_{k=1}^{K} P\left(G_i = k\right) P^k(Y_i). \tag{1}$$

By definition of a finite mixture model (Nagin 2005), the density f of Y is given by

$$f(y_i; \psi) = \sum_{k=1}^{K} \pi_k g_k(y_i; \Theta_k). \tag{2}$$

Here the group size $\pi_k > 0$ denotes the probability of a given subject to belong to group number k and thus

$$\sum_{k=1}^{K} \pi_k = 1.$$

Since in practice, it is difficult to constraint the π_k to be numbers between 0 and 1, we link the π_k to a set of parameters $\theta_1, ..., \theta_K$ such that

$$\pi_k = \frac{e^{\theta_k}}{\sum\limits_{k=1}^{K} e^{\theta_k}}.$$

The role of the parameters Θ_k is to describe the shape of the trajectories in group k. The model depends thus on the parameter set $\psi = (K, \theta_1, \cdots, \theta_{K-1}, \Theta_1, \cdots, \Theta_K)$.

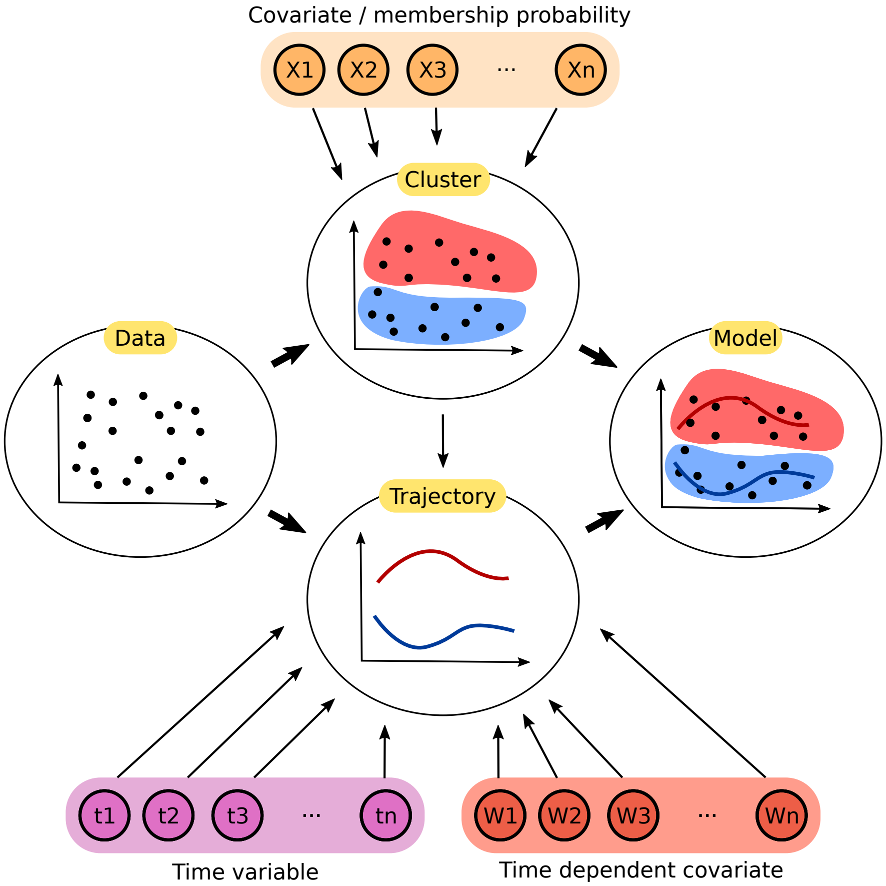
If we suppose moreover that the trajectories of Y are influenced by a static set of R risk variables $X = (X_1 \cdots X_R)$, as well as by a time-dependent covariate W which is independent of X, we can write the conditional density of Y given X and W as

$$f\left(y_i|x_i, w_i\right) = \sum_{k=1}^{K} \left(P\left(G_i = k|X_i = x_i\right) \prod_{t=1}^{T} P\left(Y_{i_t} = y_{i_t}|X_i = x_i, W_i = w_i, G_i = k\right) \right), \tag{3}$$

which can be written as

$$f\left(y_i|x_i,w_i\right)=\sum_{k=1}^{K}\left(\sum_{j=1}^{R}\frac{e^{x_i^j\theta_k^j}}{1+e^{x_i^j\theta_k^j}}\prod_{t=1}^{T}P\left(Y_{i_t}=y_{i_t}|W_i=w_i,G_i=k\right)\right). \qquad (4)$$

The generalized finite mixture model can hence be represented by the following graph:



Graphical representation of the model

The **R** package **trajeR** allows to calibrate this model for different types of densities for Y. The package can indeed currently handle the following families of densities:

- Logit
- Censored Normal
- Zero Inflated Poisson.

We generally suppose that the trajectories follow a polynomial shape, but in case of an underlying normal distribution, we include also the possibility of specifying any particular function for the shape of the trajectories, thus allowing for exponential functions for example.

A complete description of the model and the proofs of all used algorithms inside this package can be found in Noel's PhD thesis (Noel 2023). Basically, **trajeR** maximizes the log-likelihood by two methods : direct optimization of the derivative of the log-likelihood or the Expected Maximization Algorithm. For the direct optimization we use either the `optim` or the `ucminf` command. The Expected Maximization Algorithm is best suited to cases where we need to compute the Hessian matrix, whereas direct optimization of the derivative of the log-likelihood works very well if there is no Hessian matrix to compute, unless it is used together with the `ucminf` command. In both cases, we use the quasi-Newton method BFGS, which calculates an iterative approximation of the inverse of the Hessian matrix. This is useful if the derivative of the function has sharp points or if the Hessian matrix is difficult to compute or becomes numerically singular. The function `optim` is used when we do not need to store the Hessian matrix otherwise **trajeR** uses the package **ucminf** instead (Nielsen 2000).

This package is written in **R** and **C++** (Skinner 1992). The linkage between **R** and **C++** is achieved through the packages **Rcpp** (Eddelbuettel & François 2011) and **RcppArmadillo** (Eddelbuettel & Sanderson 2014).

3 Package design

The package **trajeR** is built around the core function `trajeR` which fits the model and computes its parameters for given degrees of the polynomial trajectories in the different groups. The function signature for `trajeR` is

```
R> trajeR(Y, A, Risk = NULL, TCOV = NULL, degre, degre.nu = 0,
+         Model, Method = "L",
+         ssigma = FALSE, ymax = max(Y) + 1, ymin = min(Y) - 1,
+         hessian = TRUE, itermax = 100, paraminit = NULL,
+         ProbIRLS = TRUE, refgr = 1,
+         fct = NULL, diffct = NULL, nbvar = NULL, nls.lmiter = 50)
```

Some of these arguments are mandatory others optional.

The mandatory arguments are the main data matrices `Y`, `A`, as well as `degre`, `Model` and `Method`.

Here `Y` is the matrix containing the values of the variable of interest and `A` is the matrix containing the age or time variable. In most applications, this matrix just contains times of measurement that are the same for each individual in the sample, implying that all lines of the matrix `A` are equal, but this is not necessarily the case. `A` can for instance contain the age of the different individuals at the times of measurement, which is generally different for each individual in the sample.
`degre` is a vector indicating the degree of the polynomials describing the typical trajectories in the different groups. Implicitly, the dimension of this vector also determines the number of groups into which we want to divide the population,
`Model` is a string defining the underlying distribution used in the model. The possible choices are LOGIT for the Logistic Regression Mixture Model, CNORM for the Censored Normal Mixture Model or ZIP for the Zero Inflated Poisson Mixture model.
`Method`, finally, is a string to decide which algorithm is used for estimating the model parameters. The possible choices are `L` for direct optimization, `EM` for the Expectation Maximization algorithm with quasi-Newton procedures (for LOGIT and ZIP models) and `EMIRLS` for the Expectation Maximization algorithm using Iterative Weighted Least Squares.

The optional arguments are Risk, TCOV, degre.nu, ssigma, ymax, ymin, hessian, itermax, paraminit, ProbIRLS, refgr, fct, diffct, nls.lmiter, ng.nl and nbvar.

Risk is a data matrix that contains the values of the covariate X modifying the group membership probability. By default, there is no such variable and Risk is a one-column matrix with value 1.

ProbIRLS allows to decide which method is used to compute the predictor probabilities. If its value is TRUE (default setting) we use the IRLS method and if it is FALSE we use the optimization method.

TCOV is an optional data matrix containing a time-dependent covariate W that influences the trajectories themselves. By default its value is NULL.

To ensure the identifiability of the parameters of the predictor, we have to fix a reference group. This can be done by the refgr command. It's default value is 1.

hessian indicates if we want to calculate the Hessian matrix, the default value being FALSE. If the method used is direct optimization, the Hessian matrix is computed by inverting the Fisher Information Matrix. If the method is EM or EMIRLS, the Hessian matrix is computed by using the Louis method (Louis 1982).

itermax gives the maximal number of iterations for the optim function or for the EM algorithm. The choice of the initial parameters is very important in optimization problems. We can specify these initial parameters by paraminit. By default trajeR calculates the initial value based on the range or the standard deviation of the data (for the details, see Noel (2023)).

In case of a Zero Inflated Poisson model, we have to specify the probability to belong to the excess zero state. This is done by using a polynomial logistic regression. degre.nu is the degree of the polynomial.

In case of a Censored Normal model, we have to define several arguments. ssigma indicates if we suppose to have the same standard deviation for the error terms in all groups. By default, its value is FALSE. ymax indicates the maximum of Y. It concerns only the model with censored data. By default its value is the maximum value of the data plus 1, i.e the model used is simply the normal model. Likewise, ymin indicates the minimum of Y.

There are also several arguments to define in order to use general nonlinear functions for the typical trajectories in the different groups. fct gives the definition of the function used to define the shape of the trajectories and its differential is defined in diffct. We also need to specify the number of groups to use nl.ng and the number of parameters to be estimated nbvar.

The output of trajeR is an object of class Trajectory that can be of four types depending on the model used: Trajectory.LOGIT, Trajectory.CNORM, Trajectory.ZIP or Trajectory.NL. These classes are described in the following sections.

4 The LOGIT model

For the finite mixture model with underlying binary logit distribution, called here the LOGIT model, we consider a latent variable y^*_{it} such that

$$y^*_{it} = f(a_{it}; \beta_k, \delta_k) + \varepsilon_{it} = \beta_k A_{it} + \delta_k W_t + \varepsilon_{it}, \tag{5}$$

where $\varepsilon_{it} \sim \mathcal{N}(0,\ \sigma_k)$, $A_{it} = (1, a_{it}, a_{it}^2, \cdots, a_{it}^{n_\beta - 1})^t$, $W_t = (w_{i1}, \cdots, w_{in_\delta})^t$, $\beta_k = (\beta_{k1}, \cdots, \beta_{kn_\beta})$ and $\delta_k = (\delta_{k1}, \cdots, \delta_{kn_\delta})$. n_δ denotes the dimension of the covariate W and n_β the number of measurements of Y for every individual. Here, and in the rest of this paper, we take into account that in some applications the data are actually given as a function of the age a_{it} of the subjects at the time of measurement.
The usual assumption is that the binary variable $y_{it} = 1$ if $y_{it}^* > 0$ and $y_{it} = 0$ if $y_{it}^* \leq 0$.

The probit function $\rho_{ikt} = P(Y_{it} = 1 | W_i = w_i, G_i = k)$ denotes the probability of $y_{it} = 1$ given membership in group k. We have

$$\rho_{ikt} = \frac{e^{\beta_k A_{it} + \delta_k W_{it}}}{1 + e^{\beta_k A_{it} + \delta_k W_{it}}}. \tag{6}$$

Thus,

$$g_k(y_i; \beta_k, \delta_k) = \prod_{t=1}^{T} \left(\frac{e^{\beta_k A_{it} + \delta_k W_{it}}}{1 + e^{\beta_k A_{it} + \delta_k W_{it}}} \right)^{y_{it}} \left(\frac{1}{1 + e^{\beta_k A_{it} + \delta_k W_{it}}} \right)^{1 - y_{it}} \tag{7}$$

and the log likelihood of the model is given by

$$l(\psi; y) = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} \sum_{j=1}^{R} \frac{e^{x_i^j \theta_k^j}}{1 + e^{x_i^j \theta_k^j}} \prod_{t=1}^{T} \left(\frac{e^{\beta_k A_{it} + \delta_k W_{it}}}{1 + e^{\beta_k A_{it} + \delta_k W_{it}}} \right)^{y_{it}} \left(\frac{1}{1 + e^{\beta_k A_{it} + \delta_k W_{it}}} \right)^{1 - y_{it}} \right). \tag{8}$$

The output of the `trajeR` function gives an object of class `trajectory.LOGIT`, which is a list containing at least the following components:

Component	Description
`beta`	a vector with the beta parameters
`delta`	a vector with the delta parameters (only exists in case of a covariate function W).
`theta`	a matrix with the theta parameters (becomes a vector in case there is no covariate function X).
`sd`	a vector with the standard deviations of the estimated parameters.
`tab`	a matrix with all estimated parameters and their standard deviations.
`model`	a string stating which model was used.
`groups`	an integer giving the number of groups.
`Names`	the names of the beta parameters (intercept, linear, quadratic...)
`Method`	a string stating which method was used.
`Size`	an integer giving the sample size.
`Likelihood`	a real number giving the likelihood of the model.
`Time`	a vector with the time points ie the first line of matrix A
`degre`	a vector with the degrees of the typical polynomial trajectories in the different groups.

4.1 Simulation example

We use the simulated dataset `LOGIT_data01` that comes with installing the package from the CRAN repository. The sample consists of 500 trajectories with 10 time-points each, the values of the variable Y being 0 or 1. The dataset consists in a simulated 3 group solution, the parameters of which are summarized in the following table

Parameters	β_{k0}	β_{k1}	β_{k2}	β_{k3}	β_{k4}	π_k
Group 1	-2.26	-0.109	0.14	-0.011		0.22
Group 2	1.291	0.811	-0.554	0.077	-0.003	0.44
Group 3	-1.99243					0.34

The variable of interest Y_i is contained in `data[,2:11]`, the time variable A_i in `data[,12:21]`, the time-dependent covariate W, which could for instance indicate the presence of a characteristics of the individual, in `data[,24:33]` and a covariate X influencing group membership probabilty in `data[,48:49]`. Hence,

`data[,2:11]` is a matrix with values 0 and 1.
`data[,12:21]` is a matrix with time points from 1 to 10.
`data[,22:23]` is a matrix with values 0 and 1.
`data[,24:33]` is a matrix with values 0 and 1.

Usually, we first plot the data to have a first impression about it,but this does not make a lot of sense here, since our variable of interest is dichotomous.

First, we use the function `trajR` to calibrate the basic finite mixture model. We run it once for each method. In both applications, we specify the number of groups to be 3 by putting 3 values into the parameter `degre`. In a real-life situation, we would of course need to find the optimal number of groups, as well as the optimal degree for the polynomial trajectories for each group by an exploratory procedure. Here, we know that we simulated data with polynomials of degree 0, 3 and 4 respectively for the three groups. Hence, `degree` is the vector $(0, 3, 4)$. Moreover, we specify `hessian`=TRUE to ask the computation of the Hessian matrix and set `Itermax` to 300 to ensure a good approximation of the parameters.

For the Likelihood method we call the `trajeR` function with parameter `Method` ="L".

```
R> # Likelihood
R> solL = trajeR(Y = data[,2:11], A = data[,12:21],
+                degre = c(0,3,4),
+                Model = "LOGIT", Method = "L", hessian = TRUE)
```

```
The output of trajeR gives the following parameter estimations:

Model : Logit
Method : Likelihood

  group   Parameter   Estimate   Std. Error   T for H0:   Prob>|T|
                                               param.=0
--------------------------------------------------------------------
      1   Intercept   -1.94514      0.11033   -17.63067          0

      2   Intercept   -1.78181       1.2588    -1.41549    0.15699
             Linear     -0.426      0.64991    -0.65547    0.51219
          Quadratic     0.1834      0.10798     1.69848    0.08948
              Cubic   -0.01211      0.00574    -2.11149    0.03478
```

```
     3   Intercept    1.74118      0.75191     2.31567    0.02062
            Linear     0.6527      0.76836     0.84947    0.39566
         Quadratic   -0.55852       0.2571    -2.17237    0.02987
             Cubic    0.08247      0.03354     2.45865    0.01398
           Quartic   -0.00338      0.00149    -2.26099     0.0238
--------------------------------------------------------------------
     1         pi1    0.41814      0.03917           0          0
     2         pi2    0.21874       0.0455    -14.2397          0
     3         pi3    0.36312      0.03533    -3.99337          0
--------------------------------------------------------------------
Likelihood : -2852.746
```

For the EM method, we call the `trajeR` function with parameter `Method` = ”EM” or `Method` = ”EMIRLS”.

```
R> #EM
R> solEM = trajeR(Y = data[,2:11], A = data[,12:21],
+                degre = c(0,3,4),
+                Model = "LOGIT", Method = "EM", hessian = TRUE,
+                itermax = 500)
R> #EMIRLS
R> solEMIRLS = trajeR(Y = data[,2:11], A = data[,12:21],
+                    degre=c(0,3,4),
+                    Model = "LOGIT", Method = "EMIRLS", hessian = TRUE,
+                    itermax = 500)
```

The output of `trajeR` gives the following parameter estimations (we added the previous estimations for the likelihood method as first column for easier comparison):

SolL		SolEM		SolEMIRLS	
parameters	sd	parameters	sd	parameters	sd
Beta 1					
-1.94514	0.11033	-1.94767	0.09785	-1.94514	0.09673
Beta 2					
-1.78181	1.25880	-1.34108	0.81569	-1.78180	1.01371
-0.42600	0.64991	-0.61486	0.49090	-0.42600	0.58510
0.18340	0.10798	0.20768	0.09134	0.18340	0.10483
-0.01211	0.00574	-0.01308	0.00517	-0.01211	0.00578
Beta 3					
1.74118	0.75191	1.75148	0.80594	1.74118	0.76282
0.65270	0.76836	0.79019	0.81425	0.65271	0.77462
-0.55852	0.25710	-0.61230	0.26768	-0.55852	0.25545
0.08247	0.03354	0.08893	0.03451	0.08247	0.03302
-0.00338	0.00149	-0.00362	0.00152	-0.00338	0.00146
Pi					
0.41814	0.03917	0.41731	0.02973	0.41814	0.02941
0.21874	0.04550	0.23236	0.02973	0.21874	0.02941
0.36312	0.03533	0.35033	0.04085	0.36312	0.04051

We see that the parameter estimations for the different methods are very similar. As a verification, we also calibrated the model with the **proc Traj** software and also found very similar results.

In a second step, we suppose that the membership probabilities depend on some covariate X. In our example, X is stored in `data[,22:23]` and has only the values 0 or 1.
We need to specify a reference group, the effect of the risk variavle is comapred to. By default, this reference group is the first group, but this can be changed by the parameter `refgr`. As before, we first calibrate the model by using the likelihood method.

```
R> solLRisk = trajeR(Y = data[,2:11], A = data[,12:21], Risk = data[,22:23],
+                    degre = c(0,3,4),
+                    Model = "LOGIT", Method = "L", hessian = TRUE,
+                    itermax = 300)
```

The output of `trajeR` gives the following parameter estimations:

```
 Model : Logit
 Method : Likelihood

  group   Parameter   Estimate   Std. Error   T for H0:   Prob>|T|
                                               param.=0
--------------------------------------------------------------------
      1   Intercept   -1.93481      0.10685   -18.10702          0

      2   Intercept   -4.48563      4.53011    -0.99018    0.32213
             Linear    0.88044      2.24991     0.39132    0.69557
          Quadratic    -0.0103      0.34631    -0.02975    0.97627
              Cubic    -0.0031      0.01681    -0.18432    0.85377

      3   Intercept    1.65457      0.71638     2.30962    0.02095
             Linear    0.49525      0.69724      0.7103    0.47755
          Quadratic     -0.492      0.23246    -2.11649    0.03435
              Cubic    0.07471      0.03044     2.45432    0.01415
          Quadratic   -0.00311      0.00136    -2.28584    0.02231
--------------------------------------------------------------------
      1    Baseline          0           NA          NA         NA

      2   Intercept   -0.99839      0.42801    -2.33262    0.01971
                 X1   -0.07154      0.33726    -0.21213    0.83201
                 X2    0.36804      0.34079     1.07994    0.28022

      3   Intercept    0.22836       0.2082     1.09683    0.27277
                 X1   -0.24236      0.22869    -1.05976     0.2893
                 X2   -0.38481      0.23851    -1.61338    0.10673
--------------------------------------------------------------------
Likelihood : -2849.035
```

We then launch the function `trajeR` with parameters `EM` and `EMIRLS` too. In this case, the convergence is quite slow.

```
R> solEMRisk = trajeR(Y = data[,2:11], A = data[,12:21], Risk = data[,22:23],
+                     degre = c(0,3,4),
+                     Model = "LOGIT", Method = "EM", hessian = TRUE,
+                     itermax = 500)
```

```
R> solEMIRLSRisk = trajeR(Y = data[,2:11], A = data[,12:21], Risk = data[,22:23],
+                         degre = c(0,3,4),
+                         Model = "LOGIT", Method = "EMIRLS", hessian = TRUE,
+                         itermax = 500)
```

The results of all methods are summarized in the following table:

SolLRisk		SolEMRisk		SolEMIRLSRisk	
par.	sd	par.	sd	par.	sd
Beta 1					
-1.93481	0.10685	-1.94620	0.17254	-1.93481	0.16177
Beta 2					
-4.48563	4.53011	-2.60317	2.20347	-4.48559	4.28713
0.88044	2.24991	-0.03748	1.11646	0.88043	2.13965
-0.01030	0.34631	0.12704	0.18039	-0.01030	0.33488
-0.00310	0.01681	-0.00956	0.00899	-0.00310	0.01631
Beta 3					
1.65457	0.71638	1.75301	0.75324	1.65458	0.74570
0.49525	0.69724	0.53393	0.73734	0.49525	0.68429
-0.49200	0.23246	-0.51353	0.24797	-0.49200	0.22824
0.07471	0.03044	0.07709	0.03209	0.07471	0.02982
-0.00311	0.00136	-0.00317	0.00141	-0.00311	0.00133
Theta - First group 0					
-0.99839	0.42801	-0.81927	0.61012	-0.99838	0.60966
-0.07154	0.33726	-0.10292	0.32779	-0.07154	0.33379
0.36804	0.34079	0.28454	0.36585	0.36804	0.37889
0.22836	0.20820	0.19350	0.27754	0.22836	0.26728
-0.24236	0.22869	-0.24076	0.23522	-0.24236	0.23082
-0.38481	0.23851	-0.38691	0.24532	-0.38481	0.23352

Finally, let's see an example, where we have some time-dependent covariate that influences the shape of the trajectories itself. We consider these effects by using the option `TCOV` in the command `trajeR`.
In our data, the time-dependant covariate is stored in `data[,24:33]` and has only the values 0 or 1. We again calibrate the model by using the three available methods.

In order to run the algorithm with sensible initial values, we use the results of the example without time covariate as parameter initialisation.

```
R> paraminitL = c(solL$theta[-1], solL$beta, #solution without time covariate
+                 0,0,0) # initial values for TCOV
```

Then we launch the function `trajeR`

```
R> solLTCOV = trajeR(Y = data[,2:11], A =data[,12:21], TCOV = data[,24:33],
+                    degre = c(0,3,4),
+                    Model = "LOGIT", Method = "L", hessian = TRUE,
+                    itermax = 300, paraminit = paraminitL)
```

We get the following result.

```
Model : Logit
Method : Likelihood

  group   Parameter   Estimate   Std. Error   T for H0:   Prob>|T|
                                               param.=0
-------------------------------------------------------------------
      1   Intercept   -2.01725      0.14515   -13.89748          0
              TCOV1    0.14188      0.17204     0.82469    0.40959

      2   Intercept   -1.84958      1.29591    -1.42724    0.15357
             Linear   -0.42661      0.68006    -0.62731    0.53048
          Quadratic    0.18422      0.11409     1.61473    0.10643
              Cubic   -0.01217      0.00608    -2.00396    0.04513
              TCOV1    0.11965      0.19659     0.60862    0.54281

      3   Intercept    1.72138      0.75409     2.28274    0.02249
             Linear    0.66141      0.77181     0.85696    0.39151
          Quadratic    -0.5608      0.25664    -2.18518    0.02892
              Cubic     0.0827      0.03316     2.49368    0.01267
          Quadratic   -0.00339      0.00146    -2.31694    0.02055
              TCOV1    0.01868      0.11427     0.16347    0.87016
-------------------------------------------------------------------
      1         pi1     0.4187      0.03863           0          0
      2         pi2    0.21781      0.04533   -14.41866          0
      3         pi3     0.3635       0.0339     -4.1711          0
-------------------------------------------------------------------
Likelihood : -2851.937
```

For the EM algorithm, in the initialization step, we have to transform the theta parameters to a probability.

```
R> paraminitEM = c(exp(solL$theta)/sum(exp(solL$theta)),
+                  solL$beta,  # solution without time covariate
+                  0,0,0) # initial values for TCOV

R> solEMTCOV = trajeR(Y = data[,2:11], A = data[,12:21], TCOV = data[,24:33],
+                     degre = c(0,3,4),
+                     Model = "LOGIT", Method = "EM", hessian = TRUE,
+                     itermax = 300, paraminit = paraminitEM)
R> solEMIRLSTCOV = trajeR(Y = data[,2:11], A = data[,12:21], TCOV = data[,24:33],
+                         degre = c(0,3,4),
+                         Model = "LOGIT", Method = "EMIRLS", hessian = TRUE,
+                         itermax = 300, paraminit = paraminitEM)
```

We get the following results:

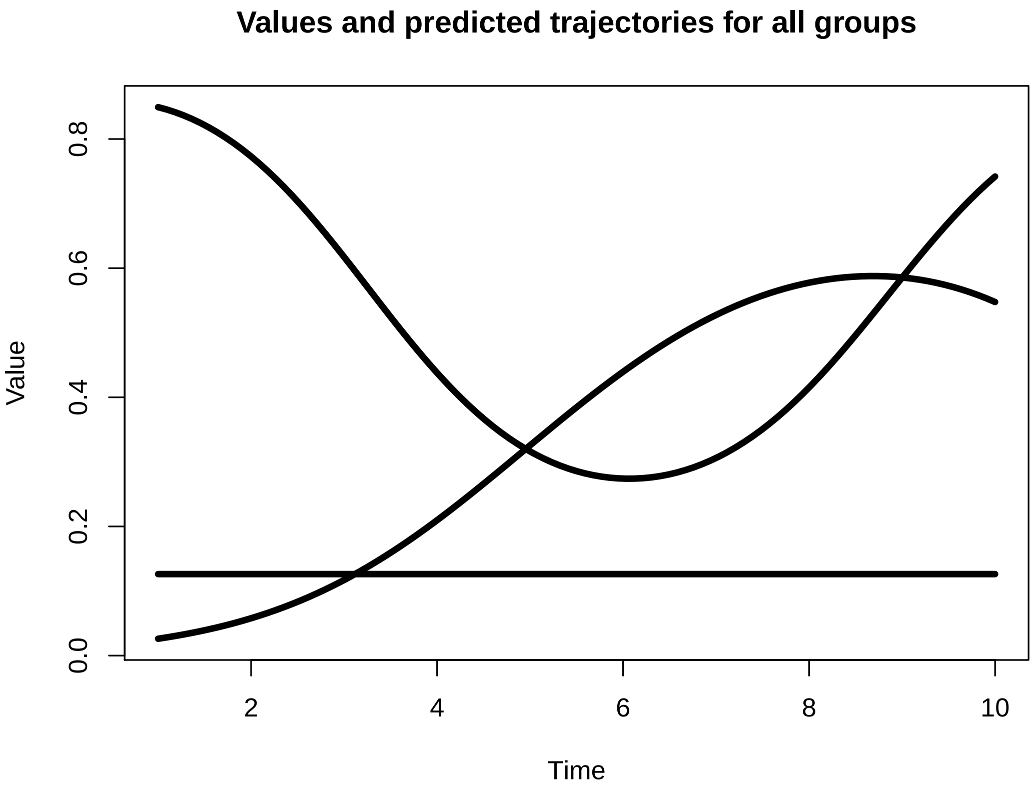
SolLTCOV		SolEMTCOV		SolEMIRLSTCOV	
par.	sd	par.	sd	par.	sd
Beta 1					
-2.01725	0.14515	-2.01680	0.13886	-2.01725	0.13877
Beta 2					
-1.84958	1.29591	-1.81020	0.99890	-1.84956	1.01962
-0.42661	0.68006	-0.44369	0.57625	-0.42662	0.58628
0.18422	0.11409	0.18646	0.10379	0.18422	0.10524
-0.01217	0.00608	-0.01226	0.00574	-0.01217	0.00581
Beta 3					
1.72138	0.75409	1.74532	0.76778	1.72138	0.76171
0.66141	0.77181	0.65570	0.77775	0.66141	0.77300
-0.56080	0.25664	-0.56069	0.25630	-0.56080	0.25495
0.08270	0.03316	0.08273	0.03312	0.08270	0.03296
-0.00339	0.00146	-0.00339	0.00146	-0.00339	0.00146
Delta 1					
0.14188	0.17204	0.14147	0.17279	0.14188	0.17277
Delta 2					
0.11965	0.19659	0.12019	0.19750	0.11965	0.19811
Delta 3					
0.01868	0.11427	0.01657	0.11479	0.01868	0.11446
Pi					
0.41870	0.03863	0.41878	0.02952	0.41870	0.02949
0.21781	0.04533	0.21898	0.02792	0.21781	0.02789
0.36350	0.03390	0.36224	0.04063	0.36350	0.04059

4.2 Plots

There are different possibilities to present the results graphically. The basic graph consists in plotting the typical trajectories of the different groups. This can be done with the command `plotrajeR` included in our **trajeR** package, applied to an object of class `Trajectory`. For instance, to plot the solution of the model with risk covariate, we simplify write

```
R> plotrajeR(solLRisk)
```

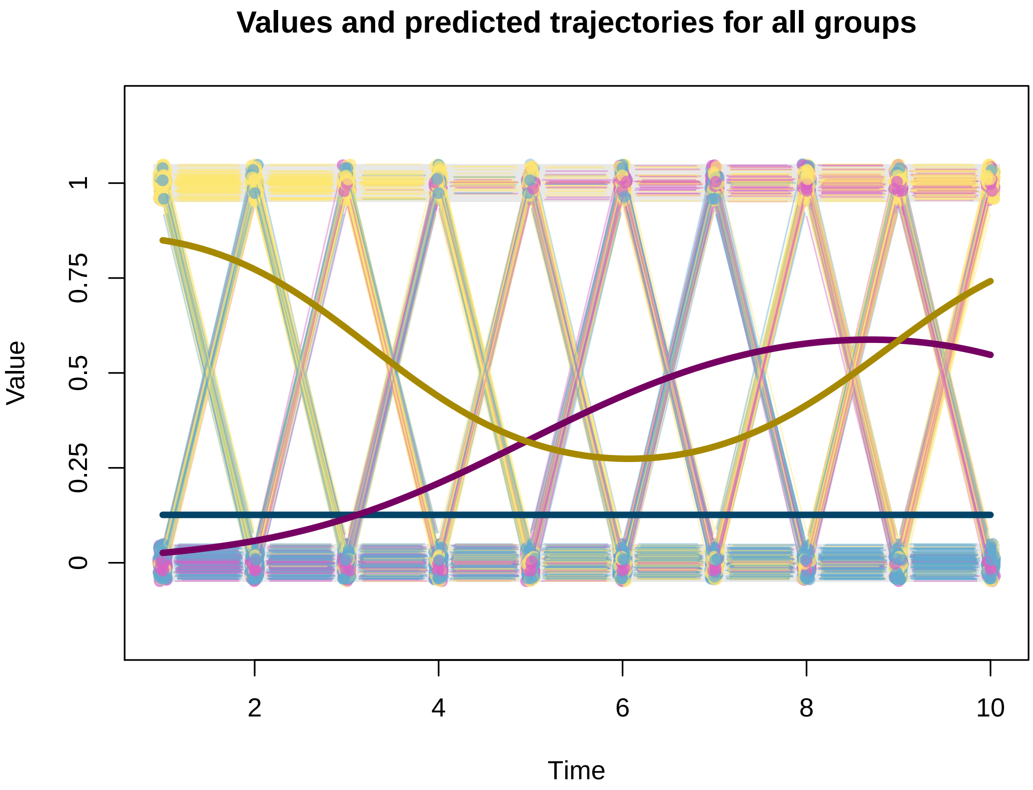
to get the following plot



We can also add the individual trajectories to the graph, with different colors for the different groups. For this, we have to specify `Y` and `A` in the function `plotrajeR()`.
For a better visibility, we use the parameter `dec` = 5, which helps to visualize the data by allowing for thicker lines in case of multiple individuals with the same trajectory.

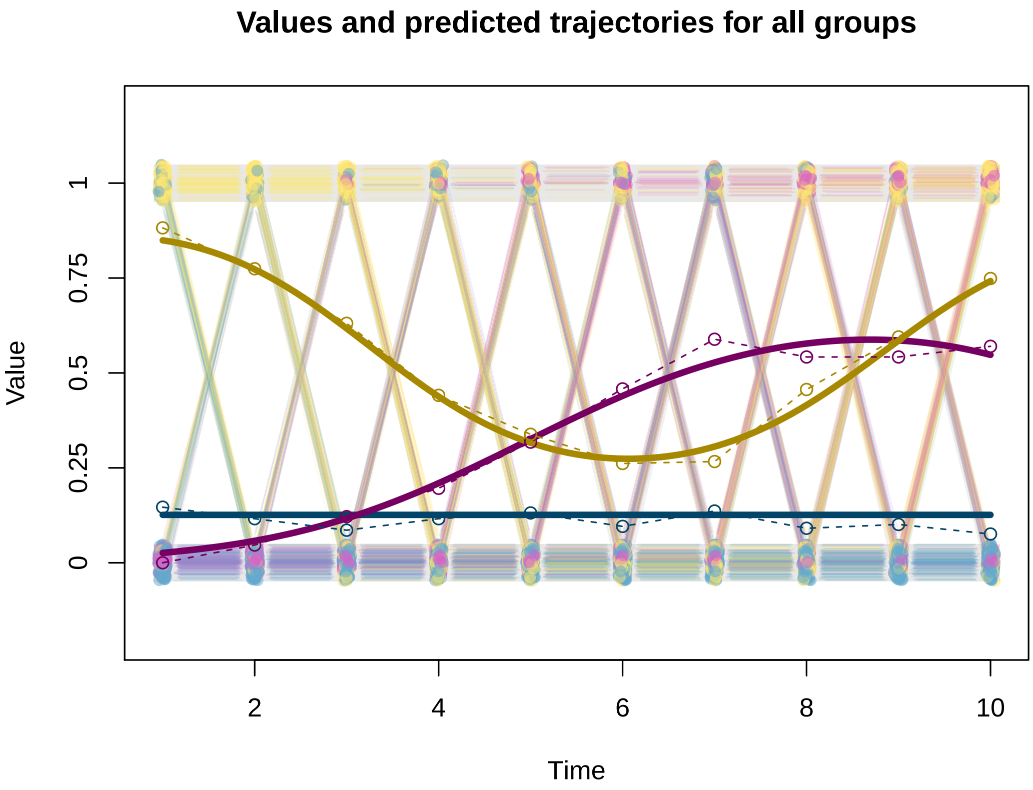
By default, the colors used for the graph are gray scale, but we can manually specify any colors we want. In that case, we need to specify two colors for each groups, one for the individual trajectories and one for the typical group trajectory.

```
R> # creation of the colors
R> trans = "70"
R> col1 = "#034569"
R> col1.1 = paste0("#64AAD0", trans)
R> col2 = "#750062"
R> col2.1 = paste0("#D962C7", trans)
R> col3 = "#A68900"
R> col3.1 = paste0("#FFE773", trans)
R> cols1 = c(col1.1, col2.1, col3.1)
R> cols2 = c(col1, col2, col3)
R> vcol = c(cols1, cols2)
R>
R> plotrajeR(solLRisk, Y = data[,2:11], A = data[,12:21], dec = 5, col = vcol)
```



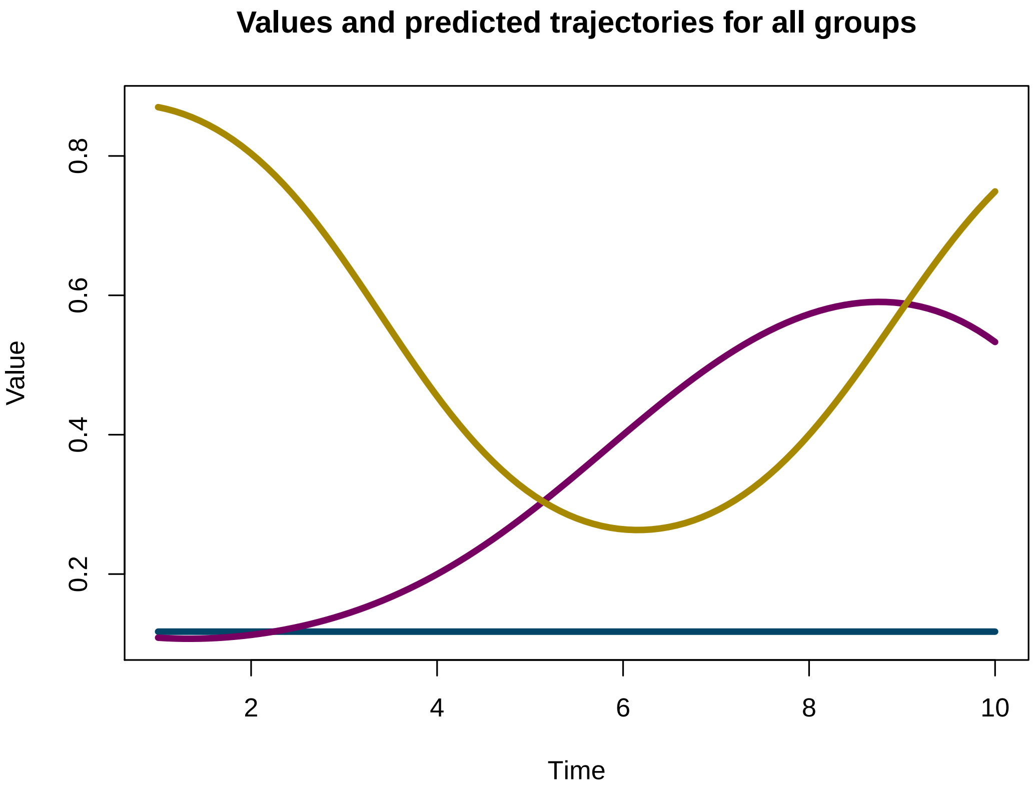
Notice that there is a difference between the typical trajectories estimated by the model for each group and the average group trajectories. The typical trajectories are indeed polynomials of degree smaller than four, whereas the average trajectories can follow any function. We can add the average of the data points on the graph by using `mean = TRUE`.

```
R> plotrajeR(solLRisk, Y = data[,2:11], A = data[,12:21], dec = 5,
+ col = vcol, mean = TRUE, alpha = 0.3)
```



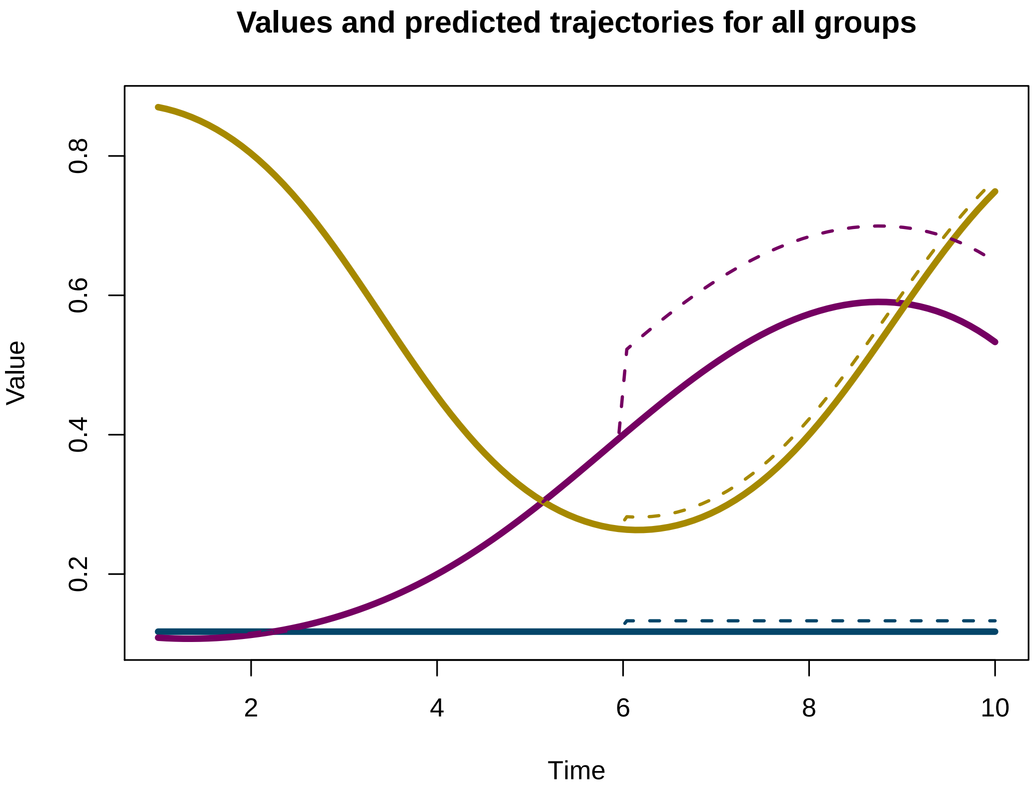
For the example with a time-dependent covariate influencing the trajectories, we get the following typical trajectories:

```
R> plotrajeR(solLTCOV, col = vcol)
```



If we want show the impact of a particular value of the time covariate on the trajectory, we can add this to the plot by using the `plotcov` parameter.
The full lines give the typical trajectories taking the time-dependent covariate into account, whereas and the dashed lines show the impact of a particular value of the covariate on the trajectories.

```
R> plotrajeR(solLTCOV, col = vcol, plotcov = c(0,0,0,0,0,1,1,1,1,1))
```



In our example, we see that the covariate has no influence before time 6. Afterwards, the covariate has only a small influence in the blue and yellow groups, but its impact in the purple group is quite big.

5 Censored Normal model

For the censored normal model, we consider a latent variable y^*_{it} such that

$$y^*_{it} = f(a_{it}; \beta_k, \delta_k) + \varepsilon_{k,it} = \beta_k A_{it} + \delta_k W_{it} + \varepsilon_{k,it}, \tag{9}$$

where $\varepsilon_{k,it} \sim \mathcal{N}(0;\ \sigma_k)$, $A_{it} = (1, a_{it}, a^2_{it}, \cdots, a^{n_\beta - 1}_{it})^t$, $W_{it} = (w_{i1}, \cdots, w_{in_\delta})^t$, $\beta_k = (\beta_{k1}, \cdots, \beta_{kn_\beta})$ and $\delta_k = (\delta_{k1}, \cdots, \delta_{kn_\delta})$.
It is easy to suppose that the variable Y is a censored variable, since the values of the data are always between two bounds y_{min} and y_{max}.
So we can link y^*_{it} to the observed and censored data y_{it} by

$$y_{it} = y_{min} \text{ if } y^*_{it} < y_{min} \tag{10}$$
$$y_{it} = y^*_{it} \text{ if } y_{min} \leq y^*_{it} \leq y_{max} \tag{11}$$
$$y_{it} = y_{max} \text{ if } y^*_{it} > y_{max}. \tag{12}$$

Denote $\mu_{ikt} = \beta_k A_{it} + \delta_k W_t$. Then

$$P(Y_{it} = y_{it} | W_i = w_i, G_i = k) = \begin{cases} \Phi\left(\frac{y_{min} - \mu_{ikt}}{\sigma_k}\right) \text{ if } y_{it}^* < y_{min} \\ \frac{1}{\sigma_k}\phi\left(\frac{y_{it} - \mu_{ikt}}{\sigma_k}\right) \text{ if } y_{min} \leq y_{it}^* \leq y_{max} \\ 1 - \Phi\left(\frac{y_{max} - \mu_{ikt}}{\sigma_k}\right) \text{ if } y_{it}^* > y_{max} \end{cases}, \tag{13}$$

where Φ and ϕ denote the cumulative distribution function and density of a standard centered normal law.

Thus, $g_k\left(y_i|;\beta_k,\delta_k,\sigma_k\right)$ becomes

$$\prod_{y_{it}=y_{min}} \Phi\left(\frac{y_{min} - \mu_{ikt}}{\sigma_k}\right) \prod_{y_{min}<y_{it}<y_{max}} \frac{1}{\sigma_k}\phi\left(\frac{y_{it} - \mu_{ikt}}{\sigma_k}\right) \prod_{y_{it}=y_{max}} \left(1 - \Phi\left(\frac{y_{max} - \mu_{ikt}}{\sigma_k}\right)\right) \tag{14}$$

and

$$l(\psi; y) = \sum_{i=1}^{n} \log\left(\sum_{k=1}^{K} \pi_k g_k(y_i; \beta_k, \delta_k, \sigma_k)\right).$$

The output of the `trajeR` function is an object of class `trajectory.CNORM`, i.e. a list containing at least the following components:

Parameters	Description
`beta`	a vector with the beta parameters.
`sigma`	a vector with the sigma parameters.
`delta`	a vector with the delta parameters (only exists if we use covariate function W).
`theta`	a matrix with the theta parameters (becomes a vector if there is no covariate function X).
`sd`	a vector with the standard deviations of the parameters.
`tab`	a matrix with all parameters and their standard deviation.
`model`	a string stating which model was used.
`groups`	a integer giving the number of groups.
`Names`	the names of the beta parameters (intercept, linear, quadratic...).
`Method`	a string stating which method was used.
`Size`	an integer giving the sample size.
`Likelihood`	a real number giving the likelihood of the model.
`Time`	a vector with the time points ie the first line of matrix A.
`degre`	a vector with the degrees of the typical polynomial trajectories in the different groups.
`min`	a real number giving the minimum of Y.
`max`	a real number giving the maximum of Y.

5.1 Simulation example

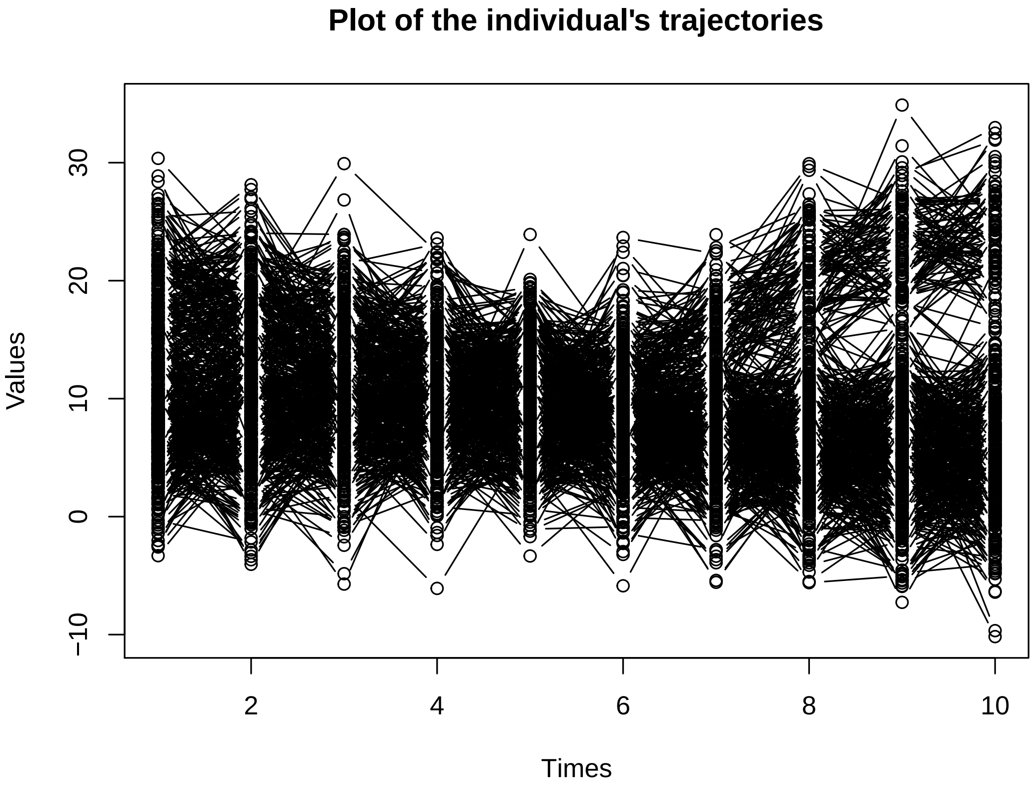
We use the simulated dataset `CNORM_data01` that comes with installing the package from the CRAN repository. The sample consists of 500 trajectories with 10 time-points each. We simulate a 3 group solution, the parameters of which are summarized in the following table

Parameters	β_{k0}	β_{k1}	β_{k2}	β_{k3}	β_{k4}	π_k	σ_k
Group 1	2.797	8.809	-3.201	0.463	-0.021	0.32	4
Group 2	7					0.54	4
Group 3	19.545	-0.297	-0.407	0.026		0.14	4

The variable of interest Y is contained in `data[,2:11]`, the time variable A in `data[,12:21]`, the time-dependent covariate W in `data[,22:41]` and a covariate X influencing group membership probability in `data[,42:43]`. Hence,

`data[,2:11]`	is a matrix with real numbers.
`data[,12:21]`	is a matrix with time points from 1 to 10.
`data[,22:31]`	is a matrix with values 0 and 1, indicating the presence or absence of a certain characteristics of the individual.
`data[,32:41]`	is a with real numbers.
`data[,42:43]`	is a matrix with values 0 and 1.

First, we plot the data to get a first impression.



Then, we start again by fitting the basic finite mixture model. We use each method to fit the model. In both applications, we specify the number of group to be 3 by putting 3 values into the parameter `degre`. We know that we simulated data with polynomials of degree 0, 3 and 4 respectively for the three groups. Hence, `degree` is the vector $(0, 3, 4)$.
Moreover, we specify `hessian`=TRUE to ask the computation of the Hessian matrix and set

`Itermax` to 300 to ensure a good approximation of the parameters.

For the Likelihood method we call `trajeR` with the option `Method ="L"`. The parameters `ssigma` allows to specify if we want the same σ in each group. `ssigma = FALSE` specifies that we have different variability in the different groups.

```
R> # Likelihood different sigma
R> solL = trajeR(Y = data[,2:11], A = data[,12:21],
+                degre = c(0,3,4),
+                Model = "CNORM", Method = "L",
+                hessian = TRUE, ssigma = FALSE)
```

We get the following result:

```
Model : Censored Normal
Method : Likelihood

  group   Parameter   Estimate   Std. Error   T for H0:   Prob>|T|
                                               param.=0
--------------------------------------------------------------------
      1   Intercept     7.0494      0.08442    83.50741          0

      2   Intercept   19.30454       0.6537    29.53102          0
             Linear   -0.09315      0.48451    -0.19225    0.84755
          Quadratic   -0.45614      0.09932     -4.5927          0
              Cubic    0.02919      0.00593     4.92296          0

      3   Intercept     1.6695      1.53075     1.09064    0.27548
             Linear   10.11827      1.73041     5.84733          0
          Quadratic   -3.70726      0.59886    -6.19052          0
              Cubic    0.53764      0.07968     6.74723          0
            Quartic   -0.02459      0.00358    -6.85989          0
--------------------------------------------------------------------
      1      sigma1    3.95795      0.05912    66.94911          0
      2      sigma2    4.11085      0.07232    56.84354          0
      3      sigma3    4.00173      0.10076    39.71375          0
--------------------------------------------------------------------
      1         pi1    0.45891      0.02837           0          0
      2         pi2    0.34901       0.0219   -12.49729          0
      3         pi3    0.19208      0.01802   -48.32612          0
--------------------------------------------------------------------
Likelihood : -14564.35
```

For the EM method we write `Method = "EM"`.

```
R> # EM
R> solEM = trajeR(Y = data[,2:11], A = data[,12:21],
+                 degre = c(0,3,4),
+                 Model = "CNORM", Method = "EM",
+                 ssigma = FALSE, hessian = TRUE)
R> solLs = trajeR(Y = data[,2:11], A = data[,12:21],
+                 degre = c(0,3,4),
+                 Model = "CNORM", Method = "L",
+                 hessian = TRUE, ssigma = TRUE)
R> solEMs = trajeR(Y = data[,2:11], A = data[,12:21],
+                  degre = c(0,3,4),
+                  Model = "CNORM", Method = "EM",
+                  ssigma = TRUE, hessian = TRUE)
```

We summarize the results in the following table:

	Different sigma				Same sigma			
	SolL		SolEM		SolLs		SolEMs	
	par.	sd	par.	sd	par.	sd	par.	sd
Beta 1								
	7.04940	0.08442	7.04940	0.08356	7.05316	0.08551	7.05316	0.08479
Beta 2								
	19.30454	0.65370	19.30454	0.60682	19.31467	0.60063	19.31467	0.59430
	-0.09315	0.48451	-0.09315	0.45447	-0.08935	0.46105	-0.08935	0.44501
	-0.45614	0.09932	-0.45614	0.09383	-0.45753	0.09708	-0.45753	0.09187
	0.02919	0.00593	0.02919	0.00563	0.02927	0.00591	0.02927	0.00551
Beta 3								
	1.66950	1.53075	1.66950	1.34917	1.66950	1.33560	1.66950	1.35545
	10.11827	1.73041	10.11827	1.52068	10.11834	1.50493	10.11834	1.52775
	-3.70726	0.59886	-3.70726	0.52667	-3.70726	0.52183	-3.70726	0.52912
	0.53764	0.07968	0.53764	0.07038	0.53764	0.06983	0.53764	0.07071
	-0.02459	0.00358	-0.02459	0.00318	-0.02459	0.00316	-0.02459	0.00320
Sigma								
	3.95795	0.05912	3.95795	0.05919	4.02028	0.06931	4.02028	0.06143
	4.11085	0.07232	4.11085	0.07048	4.02028	0.06931	4.02028	0.06687
	4.00173	0.10076	4.00173	0.09153	4.02028	0.06931	4.02028	0.09258
Pi								
	0.45891	0.02837	0.45891	0.02247	0.45978	0.02782	0.45978	0.02247
	0.34901	0.02190	0.34901	0.02151	0.34814	0.02146	0.34814	0.02150
	0.19208	0.01802	0.19208	0.03111	0.19207	0.01770	0.19207	0.03110

We see that the loglikelihood and EM methods give exactly the same parameter estimates, but the EM method is a little bit more precise in the sense that the standard deviations of the estimates are a little bit smaller for the EM algorithm.

Next, we add a risk variable influencing group membership to our model, by using the parameter `Risk`. For the likelihood method, the syntax of the command is

```
R> solLRisks = trajeR(Y = data[,2:11], A = data[,12:21], Risk = data[,42:43],
+                     degre = c(0,3,4),
+                     Model = "CNORM", Method = "L",
+                     ssigma = TRUE,hessian = TRUE)
```

In case of a common sigma for the whole population, we get

```
Model : Censored Normal
Method : Likelihood

  group   Parameter   Estimate   Std. Error   T for H0:   Prob>|T|
                                               param.=0
--------------------------------------------------------------------
      1    Intercept    7.05306      0.08474    83.23669          0

      2    Intercept   19.31517      0.58748    32.87782          0
              Linear   -0.09075      0.43826    -0.20707    0.83597
           Quadratic   -0.45723      0.09022    -5.06774          0
               Cubic    0.02926      0.00539     5.43167          0

      3    Intercept    1.66955      1.31776     1.26696    0.20523
              Linear   10.11834       1.4817     6.82887          0
           Quadratic   -3.70723      0.51392    -7.21361          0
               Cubic    0.53763      0.06885      7.8082          0
             Quartic   -0.02459      0.00312    -7.87674          0
--------------------------------------------------------------------
      1       sigma1    4.02025      0.06888    58.36946          0
      2       sigma2    4.02025      0.06888    58.36946          0
      3       sigma3    4.02025      0.06888    58.36946          0
--------------------------------------------------------------------
      1     Baseline          0           NA          NA         NA

      2    Intercept   -0.65904      0.27119    -2.43017    0.01513
                  X2    0.19531      0.35769     0.54604    0.58506
                  X3    0.57361      0.35407     1.62004    0.10529

      3    Intercept   -1.18428      0.31166     -3.7999    0.00015
                  X2     0.1063      0.41207     0.25796    0.79645
                  X3    0.52553      0.41243     1.27423    0.20264
--------------------------------------------------------------------
Likelihood : -14563.99
```

For the EM method, the syntax of the command is

```
R> solLRisks = trajeR(Y = data[,2:11], A = data[,12:21], Risk = data[,42:43],
+                     degre = c(0,3,4),
+                     Model = "CNORM", Method = "EM",
+                     ssigma = TRUE, hessian = TRUE)
```

We can summarize the different results in the following table:

Different sigma				Same sigma			
SolLRisk		SolEMRisk		SolLRisks		SolEMRisks	
par.	sd	par.	sd	par.	sd	par.	sd
Beta 1							
7.04923	0.08510	7.04923	0.08360	7.05306	0.08474	7.05306	0.08483
Beta 2							
19.30497	0.73763	19.30497	0.60687	19.31517	0.58748	19.31517	0.59428
-0.09451	0.56078	-0.09451	0.45450	-0.09075	0.43826	-0.09075	0.44498
-0.45584	0.11299	-0.45584	0.09383	-0.45723	0.09022	-0.45723	0.09186
0.02917	0.00658	0.02917	0.00563	0.02926	0.00539	0.02926	0.00551
Beta 3							
1.66955	1.41870	1.66955	1.34916	1.66955	1.31776	1.66955	1.35546
10.11828	1.59854	10.11828	1.52066	10.11834	1.48170	10.11834	1.52777
-3.70722	0.55243	-3.70722	0.52666	-3.70723	0.51392	-3.70723	0.52913
0.53763	0.07365	0.53763	0.07038	0.53763	0.06885	0.53763	0.07071
-0.02459	0.00332	-0.02459	0.00318	-0.02459	0.00312	-0.02459	0.00320
Sigma							
3.95767	0.05939	3.95767	0.05917	4.02025	0.06888	4.02025	0.06142
4.11119	0.06315	4.11119	0.07048	4.02025	0.06888	4.02025	0.06685
4.00162	0.09235	4.00162	0.09151	4.02025	0.06888	4.02025	0.09256
Theta - First group 1							
-0.65570	0.26496	-0.65570	0.26688	-0.65904	0.27119	-0.65904	0.26699
0.19567	0.35684	0.19567	0.35261	0.19531	0.35769	0.19531	0.35270
0.57547	0.35354	0.57547	0.34963	0.57361	0.35407	0.57361	0.34976
-1.18319	0.31079	-1.18319	0.32288	-1.18428	0.31166	-1.18428	0.32284
0.10679	0.40464	0.10679	0.42699	0.10630	0.41207	0.10630	0.42686
0.52683	0.39311	0.52683	0.42356	0.52553	0.41243	0.52553	0.42354

Finally, we add a time-dependent covariate by using the parameter `TCOV`. As before, we use the results of the basic finite mixture model as initial parameters. For the likelihood method, the correct syntax is

```
R> paraminit = c(solL$theta, solL$beta,solL$sigma, 0,0,0,0,0,0)
R> solLTCOV2 = trajeR(Y = data[,2:11], A = data[,12:21], TCOV = data[,22:41],
+                     degre = c(0,3,4),
+                     Model = "CNORM", Method = "L",
+                     ssigma = FALSE, hessian = TRUE)
```

The results are

```
Model : Censored Normal
Method : Likelihood

  group   Parameter   Estimate   Std. Error   T for H0:   Prob>|T|
                                               param.=0
-----------------------------------------------------------------
      1   Intercept    6.75767      0.18877    35.79904          0
```

```
            TCOV1   -0.00356      0.16759    -0.02125    0.98305
            TCOV2    0.58131      0.28646     2.02928    0.04248

      2 Intercept   19.45638       0.6439    30.21642          0
           Linear   -0.09667      0.45288    -0.21345    0.83098
        Quadratic    -0.4549      0.09361    -4.85975          0
            Cubic    0.02909      0.00562     5.17774          0
            TCOV1   -0.17069      0.19963    -0.85502    0.39258
            TCOV2   -0.13292      0.34561    -0.38458    0.70056

      3 Intercept    1.81446      1.38383     1.31119    0.18985
           Linear   10.11476      1.52332     6.63996          0
        Quadratic   -3.70808        0.527     -7.0362          0
            Cubic    0.53803      0.07039     7.64322          0
          Quartic   -0.02462      0.00318    -7.73633          0
            TCOV1     0.0805      0.26198     0.30728    0.75865
            TCOV2   -0.35937      0.46289    -0.77636    0.43757
--------------------------------------------------------------------
      1      sigma1  3.95414      0.05924    66.74877          0
      2      sigma2  4.11016      0.07047    58.32808          0
      3      sigma3  4.00019      0.09155    43.69635          0
--------------------------------------------------------------------
      1         pi1  0.45879      0.02261           0          0
      2         pi2  0.34913      0.02149   -12.71034          0
      3         pi3  0.19208      0.01784   -48.80882          0
--------------------------------------------------------------------
Likelihood : -14561.48
```

To use the EM algorithm, with and without a constant value for sigma, he write

```
R> solEMTCOV2s = trajeR(Y = data[,2:11], A = data[,12:21], TCOV = data[,22:41],
+                      degre=c(0,3,4), Model="CNORM", Method = "EM",
+                      ssigma = TRUE, hessian = TRUE)

R> solEMTCOV2 = trajeR(Y = data[,2:11], A = data[,12:21], TCOV = data[,22:41],
+                      degre=c(0,3,4), Model="CNORM", Method = "EM",
+                      ssigma = FALSE, hessian = TRUE)
```

The results are summarized in the following table:

Different sigma				Same sigma			
SolLTCOV2		SolEMTCOV2		SolLTCOV2s		SolEMTCOV2s	
par.	sd	par.	sd	par.	sd	par.	sd
Beta 1							
6.75767	0.18877	6.75768	0.18755	6.76379	0.19039	6.76375	0.19034
Beta 2							
19.45638	0.64390	19.45638	0.64707	19.46854	0.63556	19.46852	0.63333
-0.09667	0.45288	-0.09667	0.45490	-0.09328	0.44763	-0.09328	0.44524
-0.45490	0.09361	-0.45490	0.09391	-0.45624	0.09226	-0.45624	0.09191
0.02909	0.00562	0.02909	0.00563	0.02917	0.00552	0.02917	0.00551
Beta 3							
1.81446	1.38383	1.81446	1.37143	1.81446	1.35819	1.81446	1.37756
10.11476	1.52332	10.11476	1.52062	10.11485	1.50540	10.11485	1.52742
-3.70808	0.52700	-3.70808	0.52667	-3.70809	0.52211	-3.70809	0.52902
0.53803	0.07039	0.53803	0.07038	0.53803	0.06985	0.53803	0.07070
-0.02462	0.00318	-0.02462	0.00318	-0.02462	0.00316	-0.02462	0.00320
Sigma							
3.95414	0.05924	3.95414	0.05913	4.01801	0.06961	4.01801	0.06142
4.11016	0.07047	4.11016	0.07048	4.01801	0.06961	4.01801	0.06680
4.00019	0.09155	4.00019	0.09148	4.01801	0.06961	4.01801	0.09249
Delta 1							
-0.00356	0.16759	-0.00356	0.16624	-0.00763	0.16854	-0.00762	0.16875
0.58131	0.28646	0.58130	0.28559	0.58090	0.28994	0.58095	0.28982
Delta 2							
-0.17069	0.19963	-0.17069	0.19830	-0.16725	0.19367	-0.16724	0.19407
-0.13292	0.34561	-0.13291	0.34384	-0.13780	0.33793	-0.13777	0.33653
Delta 3							
0.08050	0.26198	0.08050	0.25900	0.08045	0.25954	0.08044	0.26016
-0.35937	0.46289	-0.35936	0.45630	-0.35932	0.45922	-0.35934	0.45834
Pi							
0.45879	0.02261	0.45879	0.03112	0.45970	0.02252	0.45970	0.03111
0.34913	0.02149	0.34913	0.02248	0.34823	0.02148	0.34823	0.02248
0.19208	0.01784	0.19208	0.02152	0.19207	0.01775	0.19207	0.02150

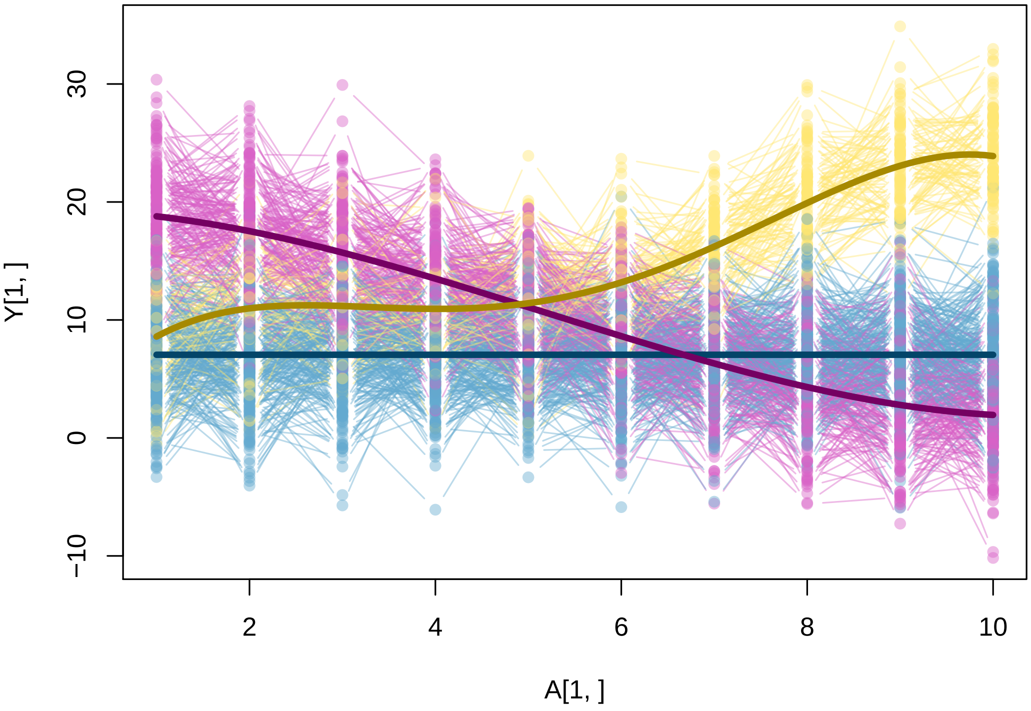
We see that the delta parameters are not significant for any of the three groups. This means that the time-dependent covariate W does actually not influence the trajectories.

5.2 Plots

We start by plotting the typical trajectories of each group for the basic finite mixture model, together with the initial trajectories of all individuals in the sample.

```
R> # creation of the colors
R> trans = "70"
R> col1 = "#034569"
```

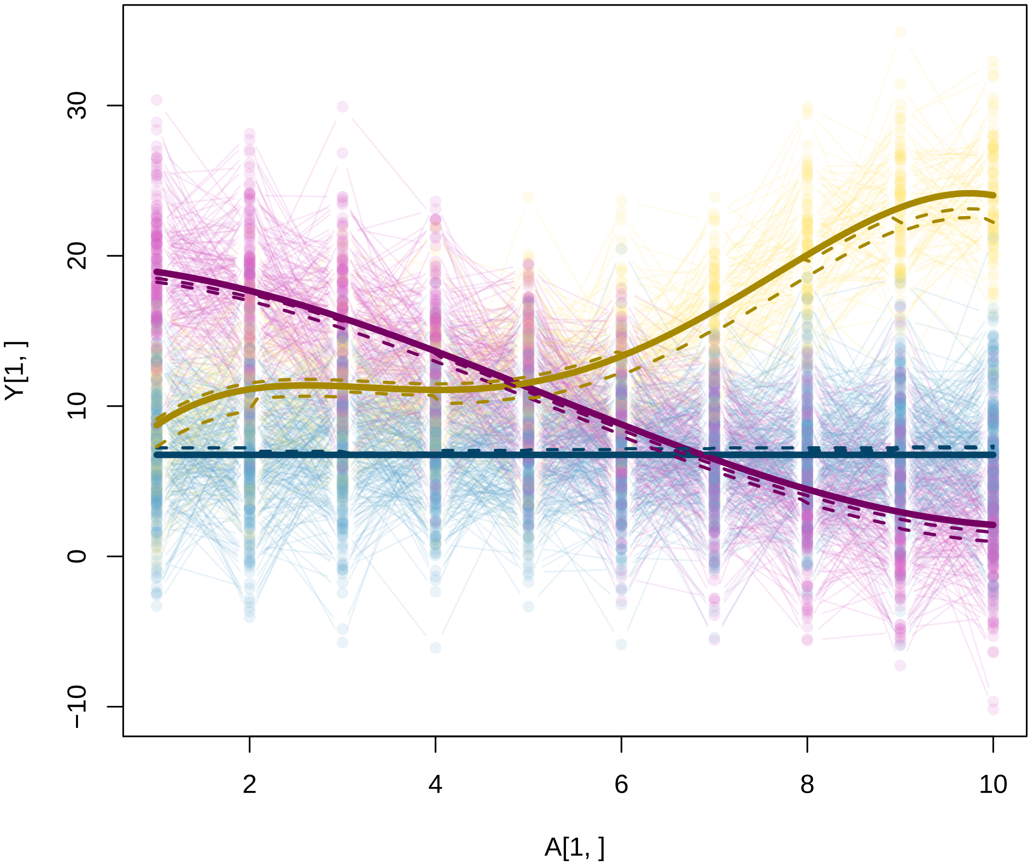
```
R> col1.1 = paste0("#64AAD0", trans)
R> col2 = "#750062"
R> col2.1 = paste0("#D962C7", trans)
R> col3 = "#A68900"
R> col3.1 = paste0("#FFE773", trans)
R> cols1 = c(col1.1, col2.1, col3.1)
R> cols2 = c(col1, col2, col3)
R> vcol = c(cols1, cols2)
R>
R> plotrajeR(solEM, Y = data[,2:11], A = data[,12:21], col = vcol)
```



We can show the effect of a time covariate on the plot by using the option `plotcov` in the function `plotrajeR`. Suppose for instance, that we want to see the effect of two different configurations of the time covariate vector. A first vector with no impact of X_1 and an X_2 with a strong impact for the first time value, then decreasing till the 3rd time-point before increasing again up to full effect in the end. And a second vector, with full impact of X_1 and an increasing effect from X_2.

```
R> trans = "25"
R> col1 = "#034569"
R> col1.1=paste0("#64AAD0",trans)
```

```
R> col2 = "#750062"
R> col2.1 = paste0("#D962C7", trans)
R> col3 ="#A68900" R> col3.1 = paste0("#FFE773", trans)
R> cols1 = c(col1.1, col2.1, col3.1)
R> cols2 = c(col1, col2, col3)
R> vcol = c(cols1, cols2)
R> plotcov = c(c(rep(0,10), 0.8,0.4,0.2,0.5,0.6,0.7,0.8,0.8,0.9,1),
                          + c(rep(1,10), 0.,0.,0.,0.,0.,0.2,0.2,0.5,0.8,0.8))
R> plotrajeR(solLTCOV2, Y = data[,2:11], A = data[,12:21],
                                    col = vcol, plotcov = plotcov)
```



5.3 Standard Normal versus Censored Normal

In this section, we illustrate the difference between a model with underlying normal distribution and underlying censored normal distribution.

The package **trajeR** actually allows to manually censor normal data by deciding on the lower and upper bound of the distribution. In the preceding example, we treated the data as coming standard normal distribution. Here, we analyze censored data and suppose that some theoretical argument tells us that our data should lie between 2 and 23. So we decide to change all values larger than 23 into 23 and all values smaller than 2 into 2.

The correponding syntax for a likelihood method is

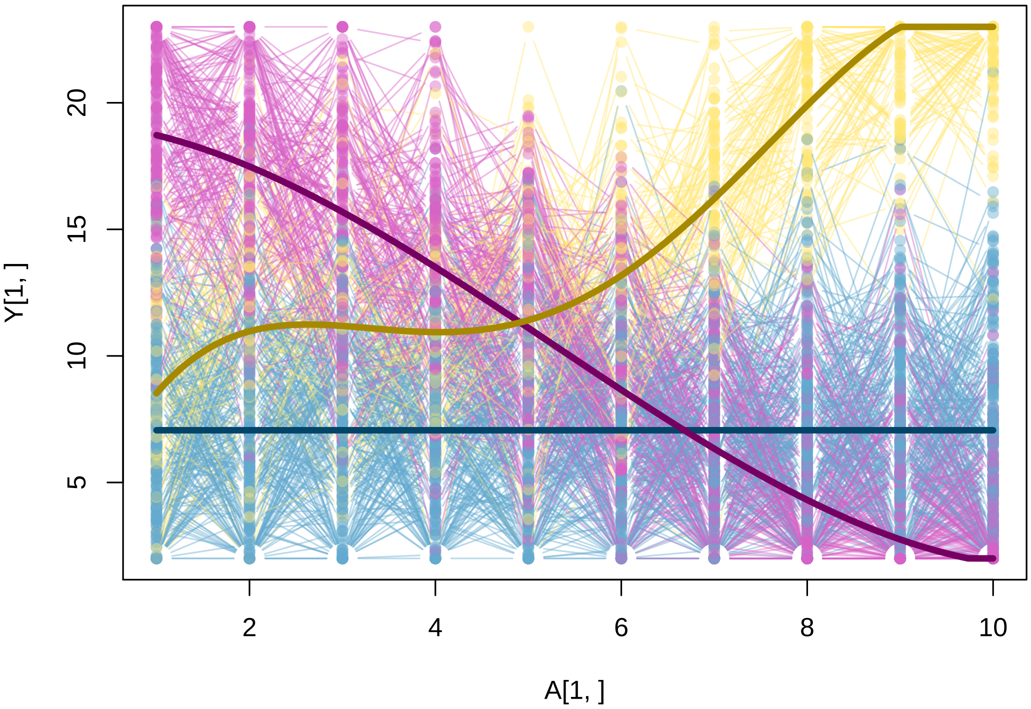
```
R> solLC = trajeR(Y = dataC[,2:11], A = dataC[,12:21],
+                 degre = c(0,3,4),
+                 Model="CNORM", Method = "L",
+                 ssigma = FALSE, hessian = TRUE,
+                 ymin=2, ymax=23)
```

The result is

```
Model : Censored Normal
Method : Likelihood

  group   Parameter   Estimate   Std. Error   T for H0:   Prob>|T|
                                               param.=0
--------------------------------------------------------------------
      1   Intercept    7.06352      0.08349    84.60614          0

      2   Intercept   19.23116      0.62361    30.83834          0
             Linear   -0.08916      0.47328    -0.18839    0.85058
          Quadratic   -0.44929      0.09893    -4.54148      1e-05
              Cubic    0.02841      0.00599     4.74365          0

      3   Intercept    1.47075      1.39933     1.05104    0.29329
             Linear   10.30212      1.58628     6.49453          0
          Quadratic    -3.7672      0.55383    -6.80215          0
              Cubic    0.54581      0.07468     7.30859          0
            Quartic   -0.02499       0.0034    -7.34021          0
--------------------------------------------------------------------
      1      sigma1    3.92236      0.06474    60.58621          0
      2      sigma2    4.10491      0.08024    51.15891          0
      3      sigma3    4.00573      0.10217    39.20749          0
--------------------------------------------------------------------
      1         pi1    0.45829      0.02795           0          0
      2         pi2    0.34961      0.02151   -12.58569          0
      3         pi3     0.1921      0.01786   -48.69404          0
--------------------------------------------------------------------
Likelihood : -13317.83
```

We see on the graph, that the typical group trajectories that leave the bounds in case of a standard normal distribution now become horizontal lines at the boundary.

```
R> plotrajeR(solLC, Y = dataC[,2:11], A = dataC[,12:21], col = vcol)
```

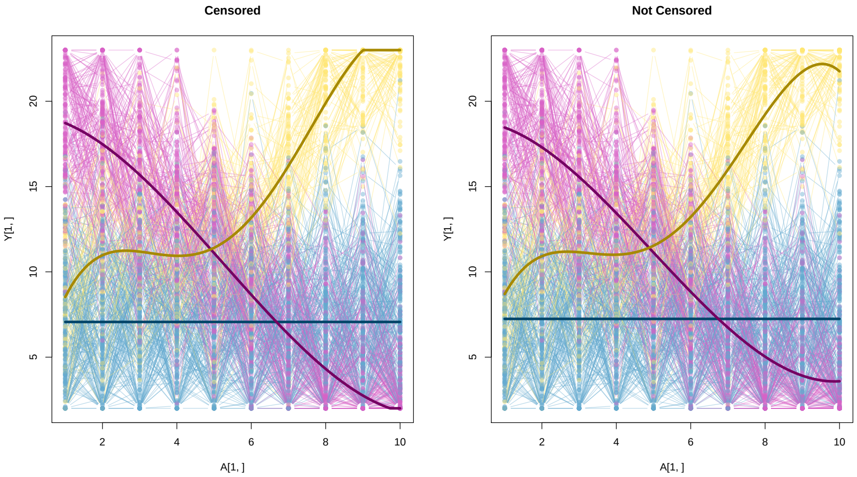


Calibrating the model without taking into account the real nature of the data leads to wrong parameter estimates. Treating a sample from a normal distribution as if it where censored data amounts to considering that the maximum and minimum of the sample are the theoretical bounds of the distribution. To illustrate this fact we compute the parameters of the previous example, without supposing that the data are censored. The result for the parameter estimates are shown in the following sample.

	Censored	Not Censored
Beta 1		
	7.0635200	7.2424800
Beta 2		
	19.2311600	18.8466500
	-0.0891600	0.0639900
	-0.4492900	-0.4835500
	0.0284100	0.0324500
Beta 3		
	1.4707500	2.3253200
	10.3021200	9.3139500
	-3.7672000	-3.4181600
	0.5458100	0.5013300
	-0.0249900	-0.0233200
Sigma		
	3.9223600	3.5980600
	4.1049100	3.5482400
	4.0057300	3.6012300
Pi		
	0.4582880	0.4588787
	0.3496098	0.3489176
	0.1921022	0.1922037

The difference between the two situations can best be seen in the graphs below. On the left side, we have the typical trajectories in case of censored data, whereas on the right side, we see the typical trajectories for an underlying normal distribution that is not censored.

```
R> par(mfrow=c(1,2))
R> plotrajeR(solLC, Y = dataC[,2:11], A = dataC[,12:21], col = vcol,
+           main="Censored")
R> plotrajeR(solLnC, Y = dataC[,2:11], A = dataC[,12:21], col = vcol,
+           main="Not Censored")
```



For censored normal data, we can of course also use the 3 methods to calibrate the model.

```
R> solLCs = trajeR(Y = dataC[,2:11], A = dataC[,12:21],
 +                   degre = c(0,3,4),
+                    Model="CNORM", Method = "L",
+                    ssigma = TRUE, hessian = TRUE,
+                    ymin = 2,  ymax = 23)
R> solEMC = trajeR(Y = dataC[,2:11], A = dataC[,12:21],
+                    degre = c(0,3,4),
+                    Model="CNORM", Method = "EM",
+                    ssigma = FALSE, hessian = TRUE,
+                    ymin = 2,  ymax = 23)
R> solEMCs = trajeR(Y = dataC[,2:11], A = dataC[,12:21],
+                     degre = c(0,3,4),
+                     Model="CNORM", Method = "EM",
+                     ssigma = TRUE, hessian = TRUE,
+                     ymin = 2,  ymax = 23)
```

	Different sigma				Same sigma			
	SolLC		SolEMC		SolLCs		SolEMCs	
	par.	sd	par.	sd	par.	sd	par.	sd
Beta 1								
	7.06352	0.08349	7.06352	0.08300	7.06261	0.08420	7.06261	0.08455
Beta 2								
	19.23116	0.62361	19.23116	0.60570	19.23109	0.59354	19.23109	0.59103
	-0.08916	0.47328	-0.08916	0.45345	-0.08164	0.44977	-0.08164	0.44246
	-0.44929	0.09893	-0.44929	0.09359	-0.45135	0.09405	-0.45135	0.09132
	0.02841	0.00599	0.02841	0.00561	0.02859	0.00569	0.02859	0.00548
Beta 3								
	1.47075	1.39933	1.47075	1.35126	1.47257	1.34574	1.47257	1.34916
	10.30212	1.58628	10.30213	1.52307	10.30028	1.52141	10.30028	1.52071
	-3.76720	0.55383	-3.76720	0.52752	-3.76648	0.53048	-3.76648	0.52670
	0.54581	0.07468	0.54581	0.07049	0.54571	0.07139	0.54571	0.07038
	-0.02499	0.00340	-0.02499	0.00319	-0.02498	0.00325	-0.02498	0.00318
Sigma								
	3.92236	0.06474	3.92236	0.06729	3.99933	0.07665	3.99933	0.07061
	4.10491	0.08024	4.10491	0.09013	3.99933	0.07665	3.99933	0.08410
	4.00573	0.10217	4.00573	0.11011	3.99933	0.07665	3.99933	0.10960
Pi								
	0.45829	0.02795	0.45829	0.02247	0.45937	0.02761	0.45937	0.02247
	0.34961	0.02151	0.34961	0.02152	0.34855	0.02138	0.34855	0.02150
	0.19210	0.01786	0.19210	0.03111	0.19208	0.01747	0.19208	0.03110

And finally, adding time-dependent covariates works in the following way.

```
R> solLCTCOV = trajeR(Y = dataC[,2:11], A = dataC[,12:21], TCOV = dataC[,22:31],
+                     degre = c(0,3,4),
+                     Model = "CNORM", Method = "L",
+                     ssigma = FALSE, hessian = TRUE,
+                     ymin = 2,  ymax = 23)
R> solLCTCOVs = trajeR(Y = dataC[,2:11], A = dataC[,12:21], TCOV = dataC[,22:31],
+                      degre = c(0,3,4),
+                      Model = "CNORM", Method = "L",
+                      ssigma = TRUE, hessian = TRUE,
+                      ymin = 2,  ymax = 23)
R> solEMCTCOV = trajeR(Y = dataC[,2:11], A = dataC[,12:21], TCOV = dataC[,22:31],
+                      degre = c(0,3,4),
+                      Model = "CNORM", Method = "EM",
+                      ssigma = TRUE, hessian = TRUE,
+                      ymin = 2,  ymax = 23)
R> solEMCTCOVs = trajeR(Y = dataC[,2:11], A = dataC[,12:21], TCOV = dataC[,22:31],
+                      degre = c(0,3,4),
+                      Model = "CNORM", Method = "EM",
+                      ssigma = TRUE, hessian = TRUE,
+                    ymin = 2,  ymax = 23)
```

	Different sigma				Same sigma			
	SolLCTCOV		SolEMCTCOV		SolLCTCOVs		SolEMCTCOVs	
	par.	sd	par.	sd	par.	sd	par.	sd
Beta 1								
	7.05899	0.11872	7.05899	0.11719	7.06103	0.12740	7.06103	0.11940
Beta 2								
	19.32126	0.63461	19.32125	0.61168	19.31862	0.61585	19.31862	0.59683
	-0.08347	0.47521	-0.08347	0.45331	-0.07567	0.45384	-0.07567	0.44232
	-0.44993	0.09884	-0.44993	0.09357	-0.45206	0.09332	-0.45206	0.09130
	0.02843	0.00597	0.02843	0.00561	0.02861	0.00561	0.02861	0.00548
Beta 3								
	1.41793	1.34024	1.41793	1.35507	1.41982	1.37305	1.41982	1.35296
	10.27971	1.50236	10.27971	1.52330	10.27787	1.54889	10.27787	1.52094
	-3.75951	0.52252	-3.75951	0.52759	-3.75879	0.53883	-3.75879	0.52677
	0.54480	0.07030	0.54480	0.07050	0.54470	0.07245	0.54470	0.07039
	-0.02494	0.00320	-0.02494	0.00319	-0.02494	0.00330	-0.02494	0.00318
Sigma								
	3.92171	0.06409	3.92171	0.06730	3.99871	0.07574	3.99871	0.07061
	4.10429	0.07937	4.10429	0.09011	3.99871	0.07574	3.99871	0.08408
	4.00512	0.10448	4.00512	0.11010	3.99871	0.07574	3.99871	0.10959
Delta 1								
	0.00793	0.16942	0.00793	0.16495	0.00200	0.17241	0.00200	0.16798
Delta 2								
	-0.20534	0.21333	-0.20533	0.19771	-0.20032	0.19041	-0.20032	0.19288
Delta 3								
	0.13640	0.28471	0.13640	0.25929	0.13626	0.26325	0.13626	0.25889
Pi								
	0.45818	0.02252	0.45818	0.03111	0.45927	0.02218	0.45927	0.03110
	0.34971	0.02164	0.34971	0.02247	0.34864	0.02166	0.34864	0.02247
	0.19210	0.01766	0.19210	0.02152	0.19209	0.01722	0.19209	0.02150

6 The Zero Inflated Poisson model

The ZIP (Zero Inflated Poisson) model consists of two separate distributions : a binary distribution that generates structural zeros, i.e. the excess zeros, and a Poisson distribution that generates the counts.

In this model, a value of zero can occur for two reasons. First, the count can be equal to zero (with a probability $P(Z_{it} = 0)$ for $Z_{it} \sim \mathcal{P}(\lambda_{ikt})$) and second the binary distribution can be zero (with a probability ρ_{ikt}).

Thus,

$$P\left(Y_{it}=y_{it}|W_i=wi,C_i=ci\right)=\begin{cases}\rho_{ikt}+(1-\rho_{ikt})e^{-\lambda_{ikt}}, & y_{it}=0\\ (1-\rho_{ikt})\frac{\lambda_{ikt}^{y_{it}}e^{-\lambda_{ikt}}}{y_{it}!}, & y_{it}>0\end{cases}. \tag{15}$$

The link functions λ_k and ρ_k that connect the developmental trajectory with age are given by

$$\log\left(\lambda_{ikt}\right)=\beta_k A_{it}+\delta_k W_t \tag{16}$$

and ρ_{ikt}

$$\log\left(\frac{\rho_{ikt}}{1-\rho_{ikt}}\right)=\nu_k A_{it}, \tag{17}$$

where $A_{it}=(1,a_{it},a_{it}^2,\cdots,a_{it}^{n_\beta-1})^t$, $W_t=(w_{i1},\cdots,w_{in_\delta})^t$, $\beta_k=(\beta_{k1},\cdots,\beta_{kn_\beta})$ and $\delta_k=(\delta_{k1},\cdots,\delta_{kn_\delta})$.

Hence,

$$g_k(y_i;\beta_k,\delta_k)=\prod_{y_{it}=0}\left(\rho_{ikt}+(1-\rho_{ikt})e^{-\lambda_{ikt}}\right)\prod_{y_{it}>0}(1-\rho_{ikt})\frac{\lambda_{ikt}^{y_{it}}e^{-\lambda_{ikt}}}{y_{it}!}$$

and the log likelihood is given by

$$l(\psi;y)=\sum_{i=1}^{n}\log\left(\sum_{k=1}^{K}\sum_{j=1}^{R}\frac{e^{x_i^j\theta_k^j}}{1+e^{x_i^j\theta_k^j}}\prod_{y_{it}=0}\left(\rho_{ikt}+(1-\rho_{ikt})e^{-\lambda_{ikt}}\right)\prod_{y_{it}>0}(1-\rho_{ikt})\frac{\lambda_{ikt}^{y_{it}}e^{-\lambda_{ikt}}}{y_{it}!}\right). \tag{18}$$

The output of the `trajeR` function is an object of class `trajectory.ZIP` that is constituted by a list containing at least the following components:

Parameters	Description
`beta`	a vector with the beta parameters.
`delta`	a vector with the delta parameters (only exists in case of a covariate function W).
`theta`	a matrix with the theta parameters (becomes a vector in case there is no covariate function X).
`nu`	a vector with the nu parameters.
`sd`	a vector with the standard deviations of the estimated parameters.
`tab`	a matrix with all estimated parameters and their standard deviations.
`model`	a string stating which model was used.
`groups`	an integer giving the number of groups.
`Names`	the names of the beta parameters (intercept, linear, quadratic...)
`Method`	a string stating which method was used.
`Size`	an integer giving the sample size.
`Likelihood`	a real number giving the likelihood of the model.
`Time`	a vector with the time points ie the first line of matrix A.
`degre`	a vector with the degrees of the typical polynomial trajectories in the different groups.
`degre.nu`	a vector with the degree of the polynomial shape for the exceeded zero state.

6.1 Simulation example

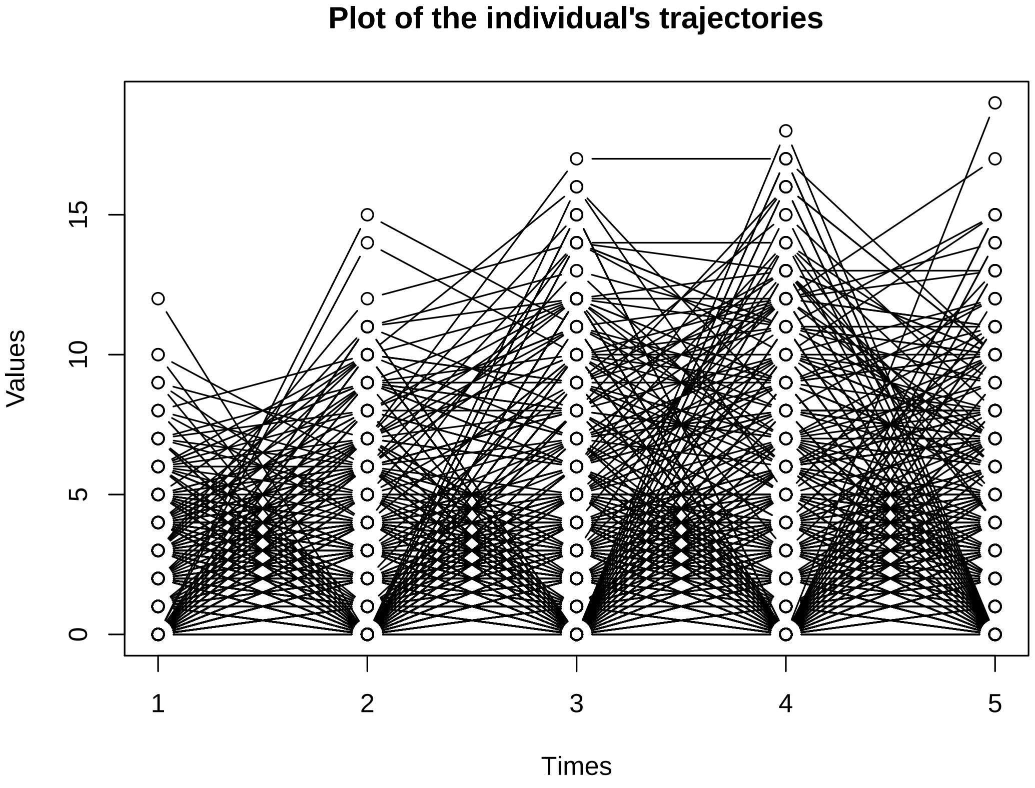
We use the simulated dataset `ZIP_data01` that comes with installing the package from the CRAN repository. The sample consists of 500 trajectories with 5 time-points each. The dataset is split into 2 groups. For group 1, the degree in the Poisson state is 2 with parameters $\beta_1 = (1.2, 0.5, -0.06)$ and for the zero state the degree is 1 with parameter $\nu_1 = (-0.2, -0.1)$. For group 2, the degree in the Poisson state is 2 with parameters $\beta_2 = (0.89, 0.01, 0.01)$ and for the zero state the degree is 1 with parameter $\nu_2 = (-1, 0)$. The parameters are summarized in the table

Parameters	β_{k0}	β_{k1}	β_{k2}	ν_{k1}	ν_{k2}	π_k
Group 1	1.2	0.5	-0.06	-0.2	-0.1	0.3
Group 2	0.89	0.01	0.01	-1	0	0.7

The variable of interest Y is stored in `dataZIP[,2:6]`, the time variable in `dataZIP[,7:11]`, the time-dependent covariate W in `dataZIP[,13:17]` and a covariate X that influences group membership in `dataZIP[,12]`. Hence,

`data[,2:6]`	is a matrix with real numbers.
`data[,7:11]`	is a matrix with integers from 1 to 5.
`data[,13:17]`	is a matrix with values 0 and 1 value.
`data[,12]`	is a matrix with real numbers.

First we plot the data to get a first impression.



We fit the model with the two methods. We choose a 2-group solution by putting 2 values in the degre and degre.nu parameters.
We specify hessian=TRUE in order to compute the Hessian matrix.

For the Likelihood method we use trajeR with parameter Method ="L".

```
R> # Likelihood
R> solL = trajeR(Y = data[,2:6], A = data[,7:11],
+               degre=c(2,2), degre.nu = c(1,1),
+               Model="ZIP", Method = "L", hessian = TRUE)
```

```
Model : Zero Inflated Poisson
Method : Likelihood

  group   Parameter   Estimate   Std. Error   T for H0:   Prob>|T|
                                               param.=0
--------------------------------------------------------------------
      1   Intercept    0.90782      0.09682     9.37679          0
             Linear    0.01872      0.07173     0.26104    0.79408
          Quadratic    0.00618      0.01157     0.53465    0.59294
               Nu11   -1.08146      0.16213    -6.67048          0
               Nu12    0.00454      0.04803     0.09451    0.92471

      2   Intercept    1.04873      0.09571    10.95765          0
             Linear    0.61507      0.06511     9.44664          0
          Quadratic   -0.07773      0.01009    -7.70501          0
               Nu21    0.03162      0.16834     0.18783    0.85102
               Nu22   -0.16776      0.05205    -3.22309    0.00128
--------------------------------------------------------------------
      1         pi1    0.65132      0.02268           0          0
      2         pi2    0.34868      0.02268    -27.5523          0
--------------------------------------------------------------------
Likelihood : -5162.009
```

For the EM method, we assign EM or EMIRLS to the parameter Method.

```
R> # EM
R> solEM = trajeR(Y = data[,2:6], A = data[,7:11],
+                degre=c(2,2), degre.nu = c(1,1),
+                Model="ZIP", Method = "EM", hessian = TRUE)
R> solEMIRLS = trajeR(Y = data[,2:6], A = data[,7:11],
+                    degre=c(2,2), degre.nu = c(1,1),
+                    Model="ZIP", Method = "EMIRLS", hessian = TRUE)
```

The results are summarized in the following table:

SolL		SolEM		SolEMIRLS	
parameters	sd	parameters	sd	parameters	sd
Beta 1					
0.90782	0.09682	0.90738	0.07865	0.90782	0.07864
0.01872	0.07173	0.01899	0.05853	0.01872	0.05853
0.00618	0.01157	0.00615	0.00947	0.00618	0.00947
Beta 2					
1.04873	0.09571	1.04879	0.07417	1.04873	0.07417
0.61507	0.06511	0.61503	0.04979	0.61507	0.04979
-0.07773	0.01009	-0.07773	0.00769	-0.07773	0.00769
Nu 1					
-1.08146	0.16213	-1.08166	0.16536	-1.08146	0.16534
0.00454	0.04803	0.00459	0.04874	0.00454	0.04874
Nu 2					
0.03162	0.16834	0.03160	0.18424	0.03162	0.18424
-0.16776	0.05205	-0.16775	0.05663	-0.16776	0.05663
Pi					
0.65132	0.02268	0.65132	0.02151	0.65132	0.02151
0.34868	0.02268	0.34868	0.02151	0.34868	0.02151

The syntax for adding a time-dependent covariate influencing the trajectories is the same than before. We again have the choice between the usual 3 methods.

```
R> solLTCOV = trajeR(Y = data[,2:6], A = data[,7:11], TCOV = data[,13:17],
+                    degre=c(2,2), degre.nu = c(1,1),
+                    Model="ZIP", Method = "L", hessian = TRUE)

R> solEMTCOV = trajeR(Y = data[,2:6], A = data[,7:11], TCOV = data[,13:17],
+                    degre=c(2,2), degre.nu = c(1,1), itermax = 500,
+                    Model="ZIP", Method = "EM", hessian = TRUE)

R> solEMIRLSTCOV = trajeR(Y = data[,2:6], A = data[,7:11], TCOV = data[,13:17],
+                         degre=c(2,2), degre.nu = c(1,1), itermax = 500,
+                         Model="ZIP", Method = "EMIRLS", hessian = TRUE)
```

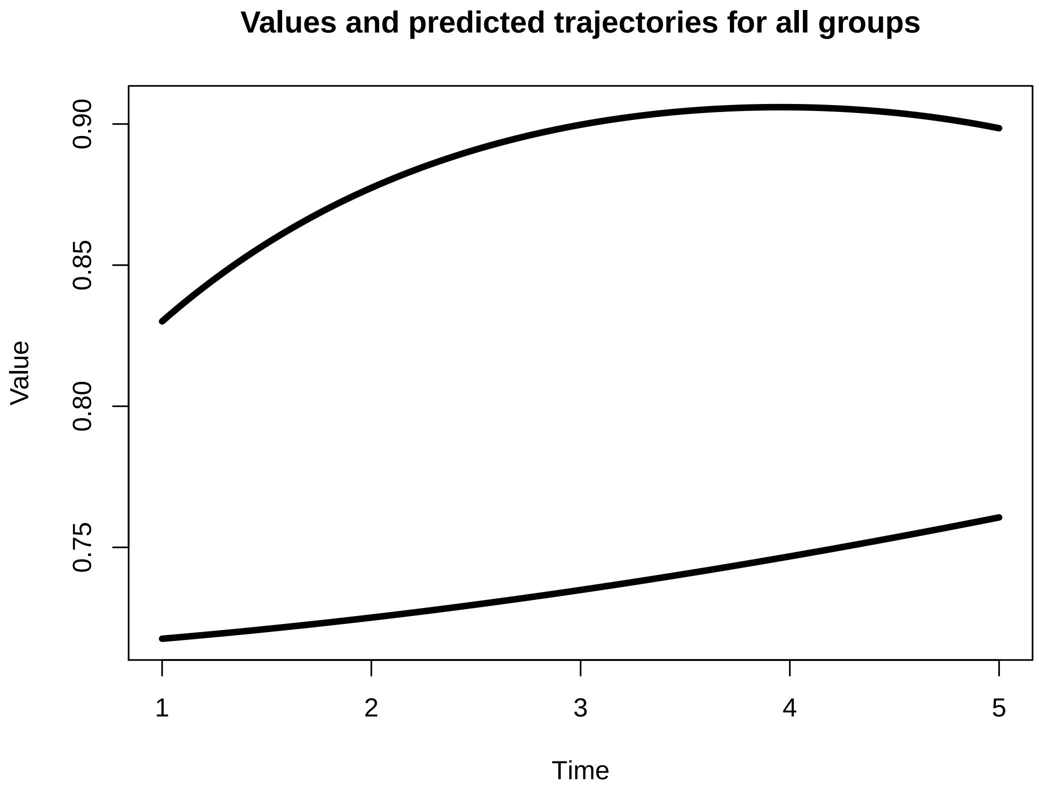
The results are summarized in the following table:

	SolLTCOV		SolEMTCOV		SolEMIRLSTCOV	
	par.	sd	par.	sd	par.	sd
Beta 1						
	0.88074	0.10032	0.88036	0.08022	0.88074	0.08022
	0.01538	0.07234	0.01557	0.05851	0.01538	0.05851
	0.00677	0.01171	0.00674	0.00947	0.00677	0.00947
Beta 2						
	1.04295	0.09688	1.04260	0.07501	1.04295	0.07500
	0.61469	0.06573	0.61492	0.04963	0.61469	0.04963
	-0.07767	0.01021	-0.07770	0.00767	-0.07767	0.00767
Nu 1						
	-1.07079	0.15832	-1.07098	0.16464	-1.07079	0.16463
	0.00234	0.04563	0.00238	0.04861	0.00234	0.04860
Nu 2						
	0.02789	0.16772	0.02785	0.18501	0.02789	0.18501
	-0.16707	0.05278	-0.16706	0.05686	-0.16707	0.05686
TCOV						
	0.06375	0.04011	0.06381	0.03164	0.06375	0.03164
	0.01297	0.03340	0.01299	0.02361	0.01297	0.02361
Pi						
	0.65198	0.02298	0.65197	0.02151	0.65198	0.02151
	0.34802	0.02298	0.34803	0.02151	0.34802	0.02151

6.2 Plots

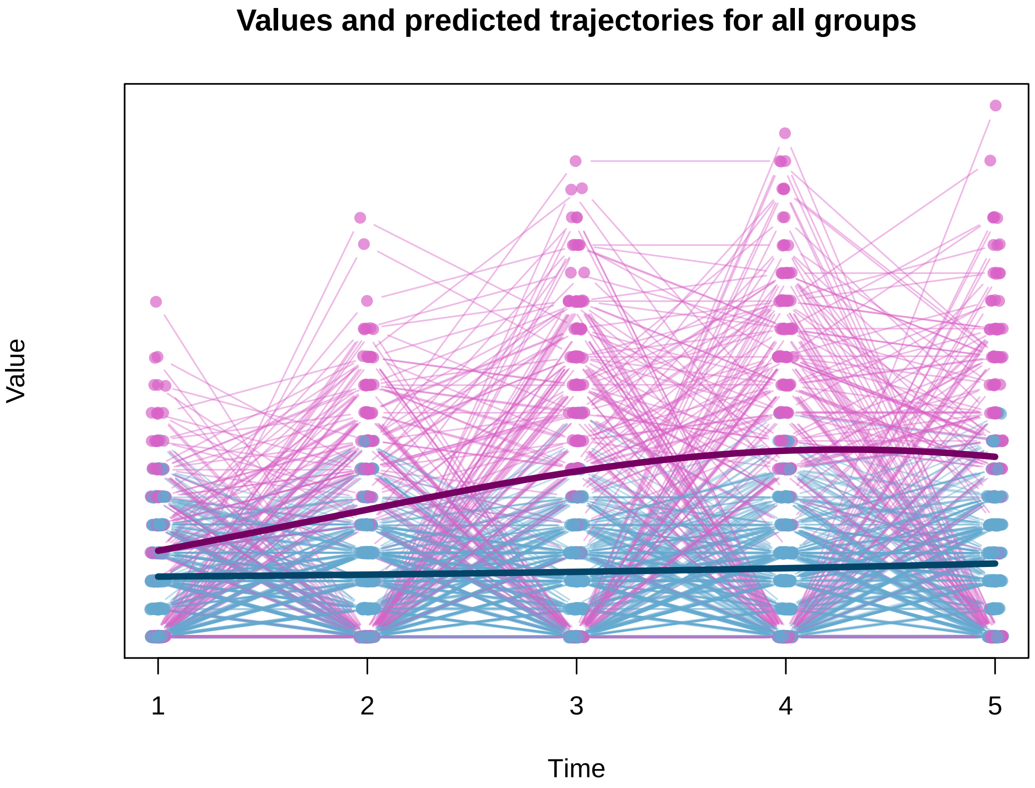
The typical trajectories of the 2 groups are plotted by

```
R> plotrajeR(solL)
```



We can also add the individual trajectories to the graph with different colors for the 2 groups.

```
R> # colour's defintion
R> trans = "70"
R> col1 = "#034569"
R> col1.1 = paste0("#64AAD0", trans)
R> col2 = "#750062"
R> col2.1 = paste0("#D962C7", trans)
R> cols1 = c(col1.1, col2.1)
R> cols2 = c(col1, col2)
R> vcol = c(cols1, cols2)
R>
R> plotrajeR(solEM, Y = dataZIP[,2:6], A = dataZIP[,7:11], dec = 4, col = vcol)
```



7 Nonlinear trajectories in case of an underlying normal distribution

For underlying normal distributions, `TrajeR` can also handle non-polynomial typical trajectories. We suppose that the variable Y_{it} is defined by

$$y_{it} = f(a_{it}; \beta_k) + \varepsilon_{it}, \tag{19}$$

where $\varepsilon_{it} \sim \mathcal{N}(0;\ \sigma_k)$.

Here, the function f can be any function depending on some parameters β_k and

$$E(Y_{it} = y_{it} | W_i = w_i, C_i = k) = f(a_{it}; \beta_k). \tag{20}$$

Then, exactly as for the censored normal model, we get

$$P(Y_{it} = y_{it} | W_i = w_i, C_i = k) = \frac{1}{\sigma_k} \phi \left(\frac{y_{it} - f(a_{it}; \beta_k)}{\sigma_k} \right). \tag{21}$$

Thus,

$$g_k(y_i;\beta_k) = \prod_{t=1}^{T} \frac{1}{\sigma_k}\phi\left(\frac{y_{it}-\mu_{ikt}}{\sigma_k}\right) \tag{22}$$

and

$$l(\psi;y) = \sum_{i=1}^{n} \log\left(\sum_{k=1}^{K} \pi_k \prod_{t=1}^{T} \frac{1}{\sigma_k}\phi\left(\frac{y_{it}-\mu_{ikt}}{\sigma_k}\right)\right). \tag{23}$$

To estimate the parameters we use the EM algorithm and the Levenberg Marquardt method.

7.1 Simulation example

We use the simulated dataset `DataNonLinear01` that comes with installing the package from the CRAN repository. The sample consists of 500 trajectories with 10 time-points each. For the typical group trajectories, we consider the nonlinear function

$$f(t;\beta_k) = \frac{\beta_k t}{\beta_k + t}.$$

The dataset consists in a simulated 4 group solution, the parameters[1] of which are summarized in the following table

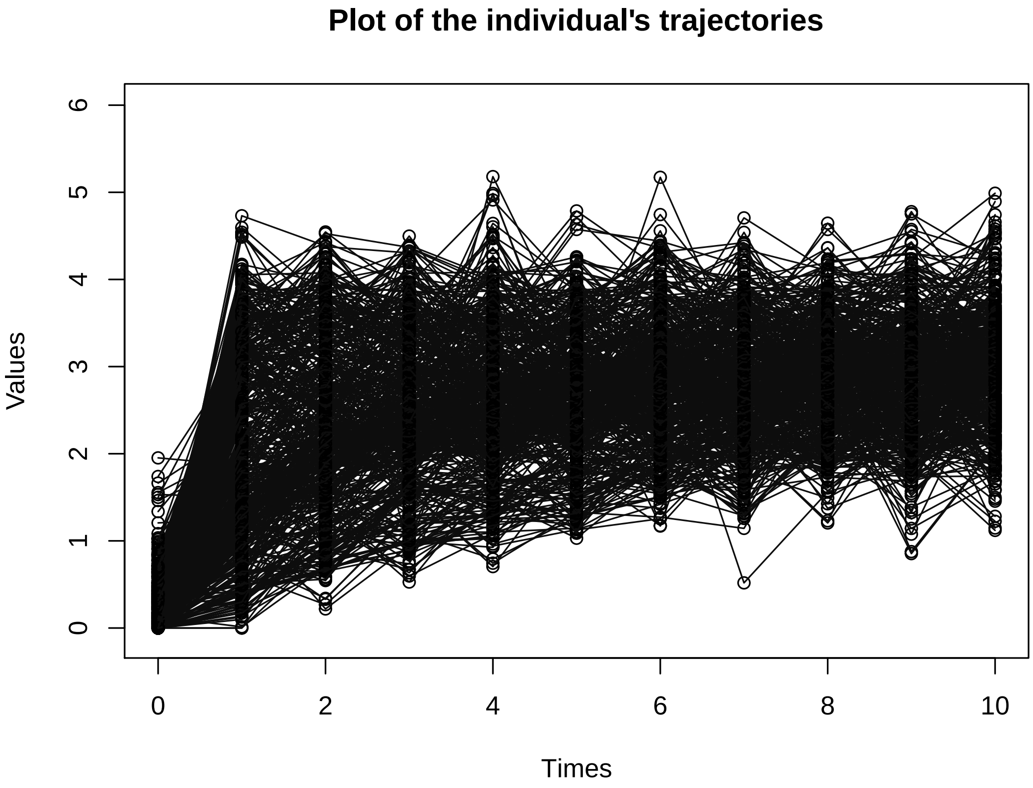
Parameters	β_{k1}	β_{k2}	π_k	σ_k
Group 1	3.6	0.025	0.45	0.2
Group 2	2.9	0.06	0.65	0.3
Group 3	3.4	5	0.35	0.14
Group 4	3.4	1.5	0.4	0.36

The variable of interest Y is stored in Y_i in `data[,2:12]`, the time variable in `data[,13:23]`. Hence,

`data[,2:12]`	is a matrix with real numbers.
`data[,13:23]`	is a matrix with integers from 0 to 10.

First we plot the data in order to get a first impression.

[1] the informed reader will have recognized, in order, *Glutamate dehydrogenase* in NAD$^+$, *Pyruvate carboxylaste* in ATP, *Threonine deaminase* in Threonine and *Hexokinase* in Fructose.



We use the EM algorithm to fit the model. We specify the number of group `ng.nl` to 4 and the number of parameters `nbvar` to 2.
`itermax` is set to 300 to ensure a good approximation of the parameters.
After analyzing the graph above, we fix the initial parameters to

```
R> paraminit=c(0.25,0.25,0.25,0.25,2,0.1,2.4,0.1,2.8,0.1,3,0.1,0.2,0.2,0.2,0.2)
```

Then we have to define the function f and its differential.

```
R> #defintion of the function
R> fct <- function(t, betak, TCOV){
+   return(
+     (betak[1]*t)/(betak[2]+t)
+   )
+ }
R> #defintion of the differential
R> diffct <- function(t, betak, TCOV){
+   return(c(
+     t/(betak[2]+t),
+     -(betak[1]*t)/(betak[2]+t)**2
+   ))
+ }
```

Finally, we call `trajeR` with parameters `Method ="EM"` and `Model="NL"`.

```
R> solEM = trajeR(Y = data[,2:12], A = data[,13:23],
+                 ng.nl = 4, nbvar = 2,
+                 Method = "EM", Model = "NL", hessian = TRUE,
+                 fct = fct, diffct = diffct, paraminit = paraminit)
```

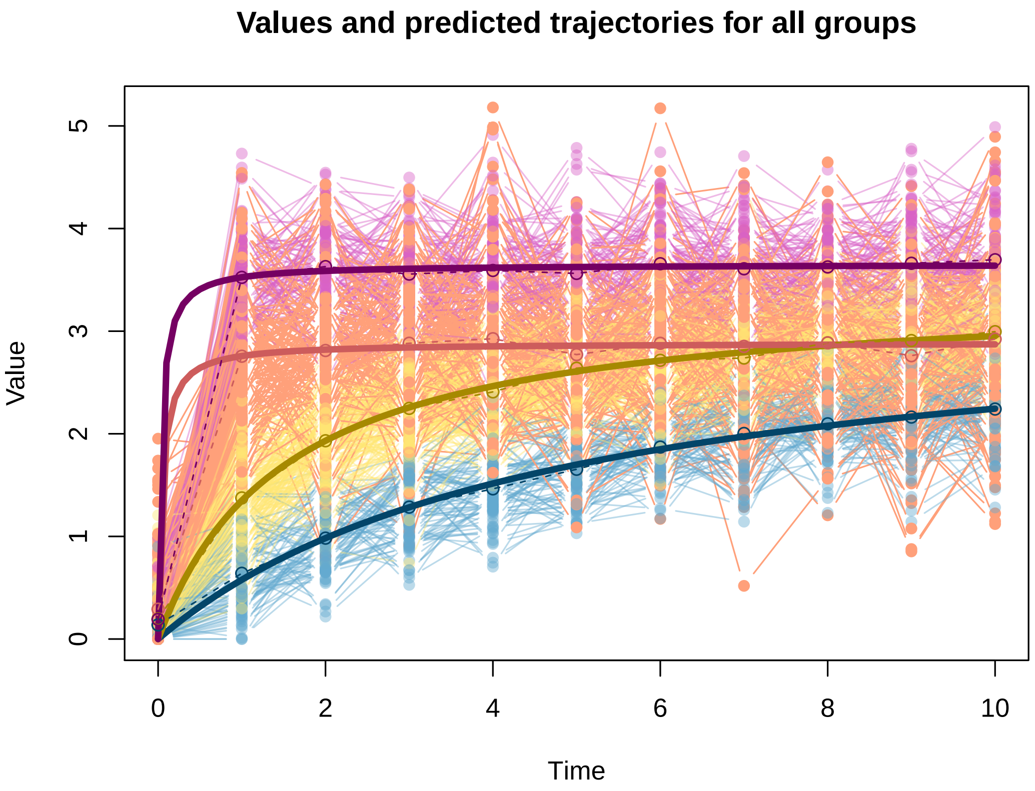
The results are given in the following table:

```
Model : Non Linear
Method : Expectation-maximization

  group   Parameter   Estimate   Std. Error   T for H0:   Prob>|T|
                                               param.=0
--------------------------------------------------------------------
      1   Intercept    3.30547       0.1011    32.69401           0
             Linear    4.72453      0.33069    14.28672           0

      2   Intercept    3.40216      0.03124   108.89768           0
             Linear    1.52195      0.05773    26.36144           0

      3   Intercept    2.88488      0.02606   110.68252           0
             Linear    0.04596      0.02516     1.82655     0.06782

      4   Intercept    3.65116      0.02378   153.51625           0
             Linear    0.03562      0.01685     2.11433     0.03453
--------------------------------------------------------------------
      1      sigma1    0.34668      0.00842     41.1552           0
      2      sigma2    0.39863      0.00725    54.95507           0
      3      sigma3    0.64933      0.01147    56.62683           0
      4      sigma4    0.43922      0.01037    42.34672           0
--------------------------------------------------------------------
      1         pi1      0.154      0.01611     9.55748           0
      2         pi2    0.36068      0.02219    16.25187           0
      3         pi3    0.31142      0.02135    14.58402           0
      4         pi4     0.1739      0.03476     5.00316           0
--------------------------------------------------------------------
Likelihood : -4209.639
```

We plot the graph with all individual trajectories, the typical group trajectories as well as the average trajectories for each group.

```
R> trans = "70"
R> col1 = "#034569"
R> col1.1 = paste0("#64AAD0", trans)
R> col2 = "#750062"
R> col2.1 = paste0("#D962C7", trans)
R> col3 = "#A68900"
R> col3.1 = paste0("#FFE773", trans)
R> col4 = "#CD5C5C"
```

```
R> col4.1 = "#FFA07A"
R> cols1 = c(col1.1, col3.1, col4.1, col2.1)
R> cols2 = c(col1, col3, col4, col2)
R> vcol = c(cols1, cols2)
R> plotrajeR(solEM, Y=data[,2:12], A=data[,13:23],col = vcol, mean = TRUE)
```



8 Some other important topics in finite mixture modeling

8.1 Group membership probabilities

It is usually very important in practice to know the group membership probabilities for all individuals in the sample. The command `GroupProb` allows to compute this. The output is a matrix of real numbers.
We illustrate this command with the results of page 22. The variable of interest Y was stored in data[,2:11], whereas the time variable was in data[,12:21]. To get the group membership probabilities of the first 8 individuals of our sample, we write

```
R> prob = GroupProb(solLs, Y =  data[,2:11], A = data[,12:21])
R> head(prob, n = 8)
```

The output is

```
              Gr1          Gr2          Gr3
[1,] 1.000000e+00 1.799942e-08 1.781949e-15
[2,] 1.931701e-07 2.276161e-16 9.999998e-01
[3,] 1.000000e+00 2.205456e-09 2.589024e-15
[4,] 1.000000e+00 2.869033e-08 6.055145e-16
[5,] 1.826708e-11 2.047939e-20 1.000000e+00
[6,] 9.999999e-01 1.116282e-07 2.165813e-13
[7,] 9.999998e-01 1.939171e-07 9.620129e-16
[8,] 9.988099e-01 1.190115e-03 6.048904e-11
```

So we see that subjects 1, 3, 4, 6, 7 and 8 are almost surely in group 1, whereas subjects 2 and 5 belong almost surely to group 3.

8.2 Group profiles

One interesting aspect of a model with covariates is the possibility to better understand the composition of the different groups by analyzing the average of the covariates in the different groups and compare them to their sample averages. These group profiles can be done using the function GroupProfiles. As an example,the syntax for getting group profiles for the 2 covariates stored in columns 42 and 43 of the matrix data, is

```
R> GroupProfiles(solLRisk, Y =  data[,2:11], A = data[,12:21], X = data[, 42:43])
```

```
          Gr 1      Gr 2      Gr 3
[1,] 0.4924891 0.5075429 0.5009375
[2,] 0.4700437 0.5187429 0.5144792
```

We see that X_1 has quite similar values in the 3 groups, whereas X_2 has a slightly lower average in group 1 than in the 2 other groups.

8.3 Model selection criteria

The criteria usually used for model selection in finite mixture models are the Bayesian Information Criterion (BIC) and the Akaike Information Criterion (AIC). These 2 classical criteria are implemented in trajeR and can be accessed by the functions trajeRBIC and trajeRAIC respectively.

```
R> trajeRBIC(solLRisk)
```

```
[1] 29255.68
```

```
R> trajeRAIC(solLRisk)
```

```
[1] 29167.18
```

8.4 Model adequacy criteria

The posterior probabilities of group membership are among other a source of valuable information for judging the model's correspondence with the data (Nagin 2005). We implemented the four diagnostics proposed in Nagin (2005) in **trajeR**.

Average Posterior Probability of Assignment

We call Average Posterior Probability of Assignment (AvePP) the average posterior probability of membership for each group for those individuals that were assigned to it. In a ideal situation the assignment probability for each individual would be 1 and the Average Posterior Probability (AvePP) would be 1 too. In our simulated example, we get

```
R> AvePP(solLRisk, Y = data[,2:11], A = data[,12:21], X = data[, 42:43])
```

```
[1] 0.9957065 0.9920113 0.9999998
```

We can see that all values are very close to 1. It means that the model fits the data correctly, which is of course not astonishing since the data were simulated to fit the model.

Odds of Correct Classification

The Odds of Correct Classification for group k ($\mathrm{OCC_k}$) is the ratio between the odds of a correct classification into group k on the basis of the posterior probability rule and the odds of correct classification based on random assignment, with the probability of assignment to group k equal to its estimated population size $\hat{\pi}_k$. Hence,

$$OCC_k = \frac{AvePP_k/(1-AvePP_k)}{\hat{\pi}_k/(1-\hat{\pi}_k}).$$

Large values of OCC_k indicate a good assignment accuracy. Nagin suggests that in a real world application an OCC_k greater than 5 for all groups is indicative that the model has a high assigment accuracy (Nagin, 2005). The function

```
R> OCC(solL, Y = data[,2:11], A = data[,12:21])
```

```
[1] 2.767407e+02 2.270867e+02 2.532774e+07
```

Estimated Group Probabilities versus the Proportion of the Sample Assigned to the Group

We compute the probability of group membership by two methods : using $\hat{\pi}_k$ or using the proportion P_k of the sample assigned to the group k. Ideally the these two values should be equal. The function `propAssign` computes P_k:

```
R> propAssign(solL, Y =data[,2:11], A = data[,12:21])
```

```
    1     2     3 
0.458 0.350 0.192 
```

We can compare with $\hat{\pi}_k$ calculated by the model.

```
R> exp(solL$theta)/sum(exp(solL$theta))
```

```
[1] 0.4589084 0.3490113 0.1920804
```

Confidence Intervals for Group Membership Probabilities

A narrow confidence interval of $\hat{\pi}_k$, for a given value of α implies that the probability is accurately estimated. These intervals are calculated by means of the bootstrap method.

```
R> ConfIntT(solL, Y = data[,2:11], A = data[,12:21], nb = 10000, alpha = 0.99)

         [,1]      [,2]      [,3]
1%  0.4399705 0.3332559 0.1828539
99% 0.4776469 0.3652074 0.2013546
```

Adequacy Matrix

trajeR allows to summarize these diagnostics in one table.

```
R> adequacy(solL,  Y = data[,2:11], A = data[,12:21], nb = 10000, alpha = 0.98)

                    1           2            3
Prob. est.   0.4589084   0.3490113 1.920804e-01
CI inf.      0.4400366   0.3333212 1.826260e-01
CI sup.      0.4770886   0.3650991 2.015266e-01
Prop.        0.4580000   0.3500000 1.920000e-01
AvePP        0.9957575   0.9918532 9.999998e-01
OCC        276.7406867 227.0867234 2.532774e+07
```

9 Summary and future plans

The **R** package **trajeR** provides all necessary tools to calibrate generalized finite mixture models, present the results graphically and test the model adequacy. In the present article we present the generalized finite mixture model for clustering time series and describe the available functions of the package. We give a step-by-step guideline on how to use the functions on simulated data sets with the three main underlying data distributions that appear in practice, the binary logit distribution, the (censored) normal distribution and the zero inflated Poisson distribution. Even though the estimation results are usually very similar, the **trajeR** package supports the two main estimations methods which are direct optimization of the derivative of the loglikelihood and the Expected Maximization Algorithm. It also allows to compute group membership probabilities for all subjects in the data set, to exhibit profiles for the clusters found in the analysis and provides a whole bunch of model adequacy criteria, together with the main model selection criteria.

We intend to develop this package in the future. Currently we work on an extension able to handle also multitrajectory models. Other foreseen extensions are to offer the beta distribution as possible underlying distribution and to take care of missing values. In medical applications for instance, it often happens that subjects disappear from the sample with time for different reasons and this needs to be taken into account in the analysis to get correct and meaningful results.

References

[1] Acock, AC. (2018). *A Gentle Introduction to Stata*. Sixth edition. College Station. Texas: Stata Press.

[2] Andruff, H., Carraro, N., Thompson, A., Gaudreau, P. & Louvet, B. (2009). *Latent Class Growth Modelling: A Tutorial*. Tutorials in Quantitative Methods for Psychology, **5-1**, 11-24.

[3] Bryk, A.S. & Raudenbush, S.W. (1992). *Hierarchical linear models*. Newbury Park, CA: Sage.

[4] Delwiche, L.D. & Slaughter, S.J. (2019). *The Little SAS Book*. Sixth edition. Cary. New Carolina: SAS Institute.

[5] Dupéré, V., Lacourse, E., Vitaro, F. & Tremblay, R.E. (2007). *Méthodes d'analyse du changement fondées sur les trajectoires de développement individuel: modèles de régression mixtes paramétriques et non paramétriques*. Bulletin de méthodologie sociologique **95**, 26-57.

[6] Eddelbuettel, C. & François, R.(2011). *Rcpp: Seamless R and C++ integration*. Journal of Statistical Software **40(8)**, 1-18.

[7] Eddelbuettel, C. & Sanderson, C.(2014). *RcppArmadillo: Accelerating R with high-performance C++ linear algebra*. Computational Statistics and Data Analysis **71**, 1054-1063.

[8] Gibbons, R.D., Hedeker D. & DuToit S. 2010). *Advances in Analysis of Longitudinal Data*. Annual Review of Clinical Psychology **6**, 79-107.

[9] Goldstein, H. (1995). *Multilevel Statistical Models*. 2nd edition. London: Arnold.

[10] Jones, B.L. & Nagin, D.S. (2007). *Advances in Group-Based Trajectory Modeling and an SAS Procedure for Estimating Them*. Sociological Methods & Research, **Vol. 35 No.4**, 542-571.

[11] Jones, B.L. Nagin, (2012). *A Stata Plugin for Estimating Group-Based Trajectory Models*. Heinz College Research Working Paper.

[12] Jones, B.L. Nagin, D.S. & Roeder, K. (2001). *A SAS Procedure Based on mixture Models for Estimating Developmental Trajectories*. Sociological Methods & Research, **Vol. 29 No.3**, 374-393.

[13] Jung, T. Wickrama, K.A.S. (2008). *An Introduction to Latent Class Growth Analysis and Growth Mixture Modeling*. Social and Personality Psychology Compass, **2-1**, 302-317.

[14] Locascio, J.J. & Atri A. (2011). *An Overview of Longitudinal Data Analysis Methods for Neurological Research*. Dementia and Geriatric Cognitive Disorders Extra **1**, 330-357.

[15] Louis, T.A. (1982). *Finding the Observed Information Matrix when Using the EM Algorithm*. Journal of the Royal Statistical Society. Series B. **44, No. 2**, 226-233.

[16] McLachlan, G. & Peel, D. (2000). *Finite mixture models*. New York: Wiley.

[17] Muthén, B.O. (1989). Latent Variable Modeling in Heterogeneous Populations. Psychometrika, **54-4**, 557-585.

[18] Muthén, B.O. & Shedden, K. (1999). Finite mixture modeling with mixture outcomes using the EM algorithm. Biomectrics, **55**, 463-469.

[19] Nagin, D.S. (1999). *Analyzing Developmental Trajectories: Semi-parametric. Groupe-based Approach*. Psychological Method, **4**, 139-157.

[20] Nagin, D.S.(2002). *Analyse des trajectoires de développement: vue d'ensemble d'une méthode semiparamétrique fondée sur le groupement*. Recueuil du Symposium de Statistique Canada Modélisation des données d'enquête pour la recherche sociale et économique.

[21] Nagin, D.S. (2005). *Group-Based Modeling of Development*. Cambridge. Massachusets: Harvard University Press.

[22] Nagin, D.S. & Land, K.C. (1993). *Age, criminal careers and population heterogeneity: Specifiction and estimation of a nonparametric, mixed Poisson model*. Criminology, **31**, 327-362.

[23] Nagin, D.S. & Odgers, C.L. (oldstylenums2010). *Group-Based Trajectory Modeling (Nearly) Two Decades Later*. Journal of Quantitative Criminology, **26**, 445-453.

[24] Nesselroade, J.R. (1991). *Interindividual differences in intraindividual change*. In L.A. Collins & J.L. Horn (eds.), Best Methods for the Analysis of Change. Washington, DC: American Psychological Association. 92-106.

[25] Nielsen, H.B. (2002). *ucminf - An algorithm for unconstrained, nonlinear optimization*. Technical Report IMM-REP-2000-19. Technical University of Denmark, Lyngby, Denmark.

[26] Noel, C. (2023). *On a generalisation of Nagin's finite mixture model*. PhD Thesis. University of Luxembourg. Luxembourg.

[27] R Core Team. (2017). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing.

[28] Roeder, K., Lynch, K.G. & Nagin, D.S. (1999). *Modeling uncertainty in latent class membership: A case study in criminology*. Journal of the American Statistical Association, **94**, 766-776.

[29] Sampson, R.J. & Laub J.H. (2005). *Seductions of method: Rejoinder to Nagin and Tremblay's "developmental trajectory groups. fact or fiction?"*. Criminology, **43-4**, 905-913.

[30] Schiltz, J. (2015). *A generalization of Nagin's finite mixture model*. In: M. Stemmler, A. Von Eye & W. Wiedermann (eds.) *Dependent Data in Social Sciences Research*. New York: Springer. 107-126.

[31] Singer, J.D. & Willet, J.B. (2003). Applied longitudinal data analysis: Modeling change and event occurrence. New York: Oxford University Press.

[32] Skinner, M.T. (1992). *The C++ Primer: A Gentle Introduction to C++*. Summit. New Jersey: Silicon Press.

[33] Verbeken, G. Fieuws, S. & Molenberghs, G. (2014). *The analysis of multivariate longitudinal data: a review*. Statistical Methods in Medical Research, **2014 (1)**, 42-59.

[34] von Eye, A. & Bergman, L.R. (2003). Research strategies in developmental psychopathology: Dimensional identity and the person-oriented approach. Development and Psychopathology **15**, 553-580.

[35] Willett,J.B. & Sayer, A.G. (1994). Using Covariance Structure Analysis to Detect Correlates and Predictors of Individual Change Over Time. Psychological Bulletin, **116-2**, 363-381.