Latent Class Growth Analysis and Growth Mixture Modeling using R: A tutorial for two R-packages and a comparison with Mplus.

Working paper (updated and corrected version 3; December 2020)

Klaas J. Wardenaar

Department of Psychiatry,
University Medical Center Groningen
Groningen, the Netherlands

Summary
Latent Class Growth Analyses (LCGA) and Growth Mixture Modeling (GMM) analyses are used to explain between-subject heterogeneity in growth on an outcome, by identifying latent classes with different growth trajectories. Dedicated software packages are available to estimate these models, with *Mplus* (Muthén & Muthén, 2019) being widely used[1]. Although this and other available commercial software packages are of good quality, very flexible and rich in options, they can be costly and fit poorly into the analytical workflow of researchers that increasingly depend on the open-source R-platform. Interestingly, although plenty of R-packages to conduct mixture analyses are available, there is little documentation on how to conduct LCGA/GMM in R. Therefore, the current paper aims to provide applied researchers with a tutorial and coding examples for conducting LCGA and GMM in R. Furthermore, it will be evaluated how results obtained with R and the modeling approaches (e.g., default settings, model configuration) of the used R-packages compare to each other and to Mplus.

Correspondence
K.J. Wardenaar PhD
University Medical Center Groningen
Department of Psychiatry (CC-72)
Hanzelplein 1
9713 GZ Groningen
Email: k.j.wardenaar@umcg.nl

[1] some major statistical software packages also offer the possibility to estimate LCGA and GMM: *SAS* (Proc Traj), *Stata*/GLLAMM (Stata corp, TX),

1. Introduction

Latent Class Growth Analyses (LCGA) and Growth Mixture Modeling (GMM) are widely used statistical models in social, behavioural and medical science. They can be used to identify latent subgroups, classes or clusters of individuals based on their common growth trajectories over time. This tutorial focuses on illustrating the procedures to carry out the actual analyses, and thus, assumes basic knowledge on the topics of longitudinal data analysis and linear mixed models (i.e. the distinction between fixed and random effects). Absolute beginners could start with some introductory texts on growth curve modeling (e.g., Curran et al., 2010) and LCGA/GMM (Jung & Wickrama, 2008; Ram & Grimm, 2009) before starting the actual analyses.

It is useful to look briefly at what LCGA and GMM are and how they are related. Both techniques are related and can be seen as an extended growth model. Growth models are used to quantify temporal trends or growth patterns in longitudinal data. Often, such models describe the outcome as a linear function of a time, but differently shaped functions are also possible (e.g., quadratic, logarithmic, cubic). The most restrictive type of growth model focuses on estimating the average growth, quantified by a fixed intercept and slope describing the average trend over time in the data. The resulting *fixed effect models* has the advantage of being easy to interpret and estimate, but often show suboptimal fit to the data, especially in case of substantial between-subject heterogeneity in change over time. It is possible to address this limitation by using a less restrictive growth model that accounts for between-subject heterogeneity by including random effects (e.g., a random intercept and, sometimes, random slope). This means that on top of the fixed effects (intercepts and slope) normally distributed variances around the fixed intercept slope are estimated. This means that a collection of individual growth trajectories instead of a single mean trend are estimated, accounting for between-subject heterogeneity in growth and allowing for explanation of between-subject variance (e.g., by including covariates). Such *random effect models*[2] allow for much more flexibility, explanation of both within- and between-subject variance in the outcome, and generally show better statistical fit to real-life data than fixed effect models. Some disadvantages of random effects models are that they are less parsimonious (more parameters) and harder to estimate than fixed effect models.

LCGA can roughly be seen as an extension of a fixed effect growth model, whereas GMM can be seen as an extension of a random effect growth model. Both techniques belong to the *latent class* or *mixture* family, with latent classes being identified based specifically on growth characteristics (i.e. growth model parameters). In this context, the terms ‘latent class’ and ‘mixture’ can be used interchangeably and both refer to the on the concept that the total observed data consists of a finite number of mixtures or latent (‘invisible’) classes, each of which is characterized by more homogeneity in the parameters or of the underlying model[3]. The mixture approach is used in many different statistical contexts, ranging from relatively simple (e.g., Latent Class Analysis (LCA; Lazarsfeld, 1950; Lazarsfeld & Henry, 1968), to quite complex models (e.g., Mixed Measurement Item Response Theory (MM-IRT; Mislevy & Verhelst, 1990). However, for all mixture models the optimal number of classes is estimated using roughly the same approach: models with increasing numbers of classes are estimated and the fit is compared between the subsequent models to select the model that best describes the data. This is also the preferred way in LCGA/GMM.

This paper focuses on using the R-platform (R-core Team, 2010) for estimating LCGA and GMM. R has several advantages over widely-used, commercially available software, such as Mplus. R-packages are free and all code is open-source, making the software broadly accessible and transparent. In addition, the R-platform and (many) packages are continually maintained/improved by a very active community of researchers and programmers. A downside is that the learning-curve for R is steep and a huge number of packages exist (of varying provenance and quality). The latter makes it hard to know where to start and what package to use for a given task. In addition, for less experienced analysts it can be a challenge to find out if a chosen package actually does what you want it to do. R-packages generally tend to cater to more advanced level analysts: relatively standard procedures like LCGA/GMM can be carried out, but are often not thoroughly documented. In fact, R-documentation

[2] Confusingly, may different terms are used in the literature to refer to such models, including Multilevel Models, Hierarchical Linear Models and Mixed Effects Models.

[3] For novice users it may be very useful to know from the start that in statistical jargon, the term *mixture model* characterizes models used to identify latent classes (by parametric methods), whereas the term *mixed model* refers to models that include both fixed and random effects. Confusingly, GMM with random effects can in fact be seen as a type of ‘Mixture Mixed Model’.

in general can be quite minimal or technical. Therefore, this paper aims to provide applied researchers with information and coding examples needed to use R to estimate LCGA/GMM models. The use of two user friendly R-packages that both offer some mixture growth modeling functionality as part of their options will be demonstrated. In addition, as software packages often differ in terms of the configuration/operationalization of the estimated model(s) and with regard to default settings, these factors will be compared between the packages and their effects on the results evaluated. Finally, the general consistency of obtained results across R-packages and Mplus will be evaluated. In the end, this should provide applied researchers with a clear entry point when embarking on LCGA/GMM in R, as well as coding examples for the typical model configurations.

2. Methods

2.1 The dataset

The used dataset (download here) was simulated based on a linear mixed models growth model with random intercept and slope and a dichotomous class covariate. The dataset contains data for 100 cases, each with 5 equally spaced repeated measures on a continuous outcome scale (y1-y5) and is simulated based on a linear mixed model with a random intercept and random slope. The data are simulated with a grouping variable that splits the sample into two classes of 50 subjects with differing mean growth. The simulation code is provided in **Appendix 1**. **Figure 1** shows the data and makes it clearly visible that there is considerable heterogeneity in growth, both across and between classes. Note that these data are model-driven and that real data can often contain more outliers and look more messy overall. Of course, class-membership is never known a priori.

With regard to structure, data can have a ‘long’ format, with multiple rows per subject and the number of rows per subjects corresponding to the number of repeated measurements, or a ‘wide’ format, with a single row per subject and a variable (column) for each repeated measurement of the outcome. The R-packages use the long format and Mplus uses the wide format.

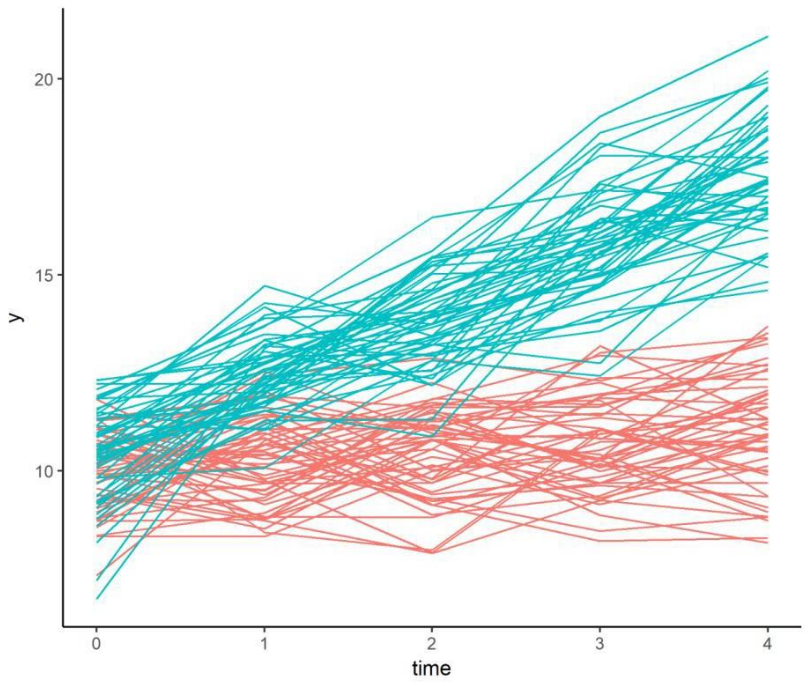

Figure 1: simulated data for use in the tutorial: the different colors denote the two simulated subgroups.

2.1 Software

There are several commercially available software packages to conduct LCGA or GMM. Probably the most widely used packages are SAS (PROC TRAJ (Jones et al., 2001; mainly for LCGA) and Mplus (Muthén & Muthén, 2013; for both LCGA and GMM). Especially Mplus offers a broad range of modeling options, covering most structural equation modeling (SEM) approaches, latent variable models and various item response theory models. Given that Mplus is so widely used for LCGA and GMM, it will be used to provide the reference models in the current paper[4].

For R, there are several packages that allow for growth modeling and/or mixture modeling. Of these, several allow users to fit LCGA and/or GMM models, of which two packages will be illustrated here in detail. First, the *'lcmm'* package offers many advanced options for mixture modeling with longitudinal data, but also provides the options needed to conduct LCGA and GMM, using the '*hlme*' function (Proust-Lima et al., 2017; https://cran.r-project.org/web/packages/lcmm/index.html). Second, the 'flexmix' package (Leisch, 2004; https://cran.r-project.org/web/packages/flexmix/index.html) provides functionality for flexible mixture regression modeling, including LCGA and GMM models. The used versions for this tutorial are Mplus 5.0, R v3.6.2, Rstudio v1.2.5033, 'lcmm' v1.8.1, and 'flexmix' v2.3-15.

2.2 Covered models and analytical approach

This tutorial illustrates the following, LCGA and GMM variants: (1) LCGA (fixed intercept and slope), (2) Growth Mixture Modeling with class-specific random intercepts (GMM-1), and (3) Growth Mixture Modeling with class-specific random intercepts and slopes (GMM-2). The estimated models will be limited to linear growth functions. Because there is always a risk of a mixture solution at a local maximum rather than a global maximum likelihood value in mixture modeling, all models are run multiple times with varying random start values to identify a replicable solution at the global maximum.

With each package, the same general analytical approach will be used. In each analysis, the first step of the process will be to define the underlying growth model. Second, models with increasing numbers of classes (1-4 classes) will be fit to the data. Third, the optimal model will be identified by comparing the Bayesian Information Criterion (BIC) and Akaike Information Criterion (AIC), with the lowest values indicating the best model given the data. Finally, the posterior class probabilities (probabilities with which each case is allocated to each class) are inspected and cases are allocated to their most probable class. The number of cases and observed/estimated trajectory per class will be inspected. The analyses with the different packages will be run with the same simulated dataset, allowing for direct comparison of the results across the packages and for comparison between the R-packages and Mplus.

2.3 Code examples

For each package (*lcmm, flexmix* and *Mplus*), example code is provided in dedicated text boxes. This code can be directly copied and pasted into R or Mplus, but users need to make sure to adjust data file names, object names, and variable names according to their specific setting. For clarity, all user-specified elements (datasets, objects and variable names) are printed in blue. All annotation throughout the code is printed in green.

[4] it is important to point out that there are other commercial packages also offer possibilities to estimate LCGA, GMM and more. These include Latent Gold (Vermunt & Magidson, 2016) and Stata (GLLAMM; Rabe-Hesketh et al., 2004).

Table 1: model fit statistics for latent class growth models (LCGA) estimated with different software packages using the same simulated dataset.

Software	Number of classes	Free parameters	LL	AIC	BIC	Class-sizes
‘lcmm’	1-class	3	-1146.3	2298.6	2306.5	100
	2-class	6	-875.5	1762.9	1778.5	50, 50
	3-class	9	-858.3	1734.6	1758.0	48, 19, 33
‘flexmix’	1-class	3	-1146.3	2298.6	2311.3	100
	2-class	6	-875.5	1762.9	1788.2	50, 50
	3-class	9	-858.3	1734.6	1772.5	19, 48, 33
Mplus[1]	1-class	3	-1146.3	2298.6	2306.5	100
	2-class	6	-875.5	1762.9	1778.5	50, 50
	3-class	9	-858.3	1734.6	1758.0	33, 48, 19

[1]This model was run while overriding the default setting to estimate a residual variance term per time point by imposing an equality constraint across time-points.
[2]BIC calculated based on total number of data points instead of total number of subjects.

Table 2: model fit statistics for growth mixture models (GMM), estimated with different software packages using the same simulated dataset.

Software		Number of classes	Free parameters	Log-likelihood	AIC	BIC	Class-sizes
lcmm	Random intercept	1-class	4	-1042.0	2091.9	2102.4	100
		2-class	8	-855.3	1726.7	1747.5	50, 50
		3-class	12	-838.7	1701.4	1732.7	35, 45, 20
	Random intercept and slope	1-class	6	-857.5	1727.1	1742.7	100
		2-class	**10**	**-828.8**	**1677.5**	**1703.6**	**50, 50**
		3-class	14	-826.0	1679.9	1716.4	18, 50, 32
flexmix	Random intercept	1-class	4	-1042.0	2091.9	2108.8	100
		2-class	8	-855.3	1726.7	1760.4	50, 50
		3-class	12	-838.7	1701.4	1752.0	35, 45, 20
	Random intercept and slope	1-class	6	-857.5	1727.1	1752.4	100
		2-class	**12**	**-828.4**	**1680.7**	**1731.3**	**50, 50**
		3-class	18	-824.6	1685.1	1761.0	22, 49, 29
Mplus[1]	Random intercept	1-class	4	-1042.0	2091.9	2102.4	100
		2-class	8	-855.3	1726.7	1747.5	50, 50
		3-class	12	-838.7	1701.4	1732.7	35, 45, 20
	Random intercept and slope	1-class	6	-857.5	1727.1	1742.7	100
		2-class	**12**	**-829.4**	**1680.7**	**1712.0**	**50, 50**
		3-class	18	-820.6	1677.1	1724.0	8, 42, 50

The lines in bold font represent the models with the best fit according to the AIC and/or BIC.
[1]This model was run while overriding the default setting to estimate a residual variance term per time point by imposing an equality constraint across time-points.
[2]BIC calculated based on total number of data points instead of total number of subjects.

3. Tutorial and Results

3.1 *Package lcmm*

3.1.1 *LCGA with lcmm*

The ‘*hlme()*’ function from the *lcmm* package can be used to run a LCGA in R, using the following code:

```
# set working directory (location of data)
set.wd(…)

# open package
library(lcmm)

# set the seed for random number generator, so results can be
# reproduced:
set.seed(2002)

# run models with 1-4 classes, each with 100 random starts,
# using the 1-class model to set initial start values:
lcga1 <- hlme(y ~ time, subject = "ID", ng = 1, data = mydata)
lcga2 <- gridsearch(rep = 100, maxiter = 10, minit = lcga1,
          hlme(y ~ time, subject = "ID",
               ng = 2, data = mydata, mixture = ~ time))
lcga3 <- gridsearch(rep = 100, maxiter = 10, minit = lcga1,
                     hlme(y ~ time, subject = "ID",
                           ng = 3, data = mydata, mixture = ~ time))

# make table with results for the 4 models:
summarytable(lcga1, lcga2, lcga3)
```

We start by opening the package and setting the seed to be used by the random number generator, which will allow us to exactly reproduce the results of the procedures with multiple random starts if we rerun the analyses. The `hlme()` function starts with a formula statement (‘`y~time`’) to define the growth model. The `subject=”ID”` statement identifies the unit (subject) within which the repeated measures are nested. The function `hlme` allows for specification of random effects (slope and intercept). In the code above, these are omitted, leading to estimation of LCGA models with a fixed intercept and slope per class. Here, `hlme` is used as a sub-function in the `gridsearch()` function. The latter is used to run each `hlme` model 100 times (`rep=100`) using different start values to avoid local maxima. Here, the start values are based on the 1-class model (`minit=lcga1`). All of the user-specified settings can of course be adjusted to fit one’s aims.
Running the code in the simulated data yields individual model objects (`lcga1-lcga3`). The `summarytable()` function can be used to tabulate all models for comparison on degrees of freedom (`npm`), indices of fit (AIC and BIC) and class sizes:

```
      G      loglik npm      AIC      BIC %class1 %class2 %class3
lcga1 1 -1146.3214   3 2298.643 2306.458     100
lcga2 2  -875.4555   6 1762.911 1778.542      50      50
lcga3 3  -858.2967   9 1734.593 1758.040      48      19      33
```

To inspect a specific model more closely, we can look at its model summary, using `summary(lcga2)`, which provides the following, more extensive, information:

```
Goodness-of-fit statistics:
     maximum log-likelihood: -875.46
     AIC: 1762.91
     BIC: 1778.54


Maximum Likelihood Estimates:

Fixed effects in the class-membership model:
(the class of reference is the last class)

                     coef      Se   Wald p-value
```

```
intercept class1 -0.00075 0.12108 -0.006 0.99509

Fixed effects in the longitudinal model:

                     coef       Se    Wald p-value
intercept class1 10.49814 0.13299 78.939 0.00000
intercept class2  9.95581 0.13297 74.872 0.00000
time class1       1.76654 0.05439 32.478 0.00000
time class2       0.26565 0.05432  4.890 0.00000

                             coef       Se
Residual standard error:  1.21375 0.03844
```

In addition to the estimation details and goodness-of-fit information, this output also shows the class-specific intercepts and slopes (i.e. '`time class1`' and '`time class2`') as well as the residual standard deviation, providing insight into the overall unexplained variance ($1.21375^2 \approx 1.39$).

3.1.2 GMM-1 with lcmm

To estimate GMMs with only a random intercept using *lcmm*, the LCGA code can be slightly adjusted as follows:

```
set.seed(2002)
gmm1 <- hlme(y ~ time, subject = "ID", random=~1, ng = 1, data =
             mydata)
gmm2 <- gridsearch(rep = 100, maxiter = 10, minit = gmm1,
           hlme(y ~ time, subject = "ID", random=~1,
                ng = 2, data = mydata, mixture = ~ time,
                    nwg=T))
gmm3 <- gridsearch(rep = 100, maxiter = 10, minit = gmm1,
                    hlme(y ~ time, subject = "ID", random=~1,
                        ng = 3, data = mydata, mixture = ~ time,
                        nwg=T))

# make table with results for the 4 models:
summarytable(gmm1, gmm2, gmm3)
```

The main difference with the LCGA code lies in the addition of the '`random=~1`' part to the model definition. This indicates that we want to include a random intercept per class: i.e. a normal distribution of individual intercepts around a fixed mean level. The '`nwg=T`' statement indicates that we want the variances of the random intercepts to be estimated separately for each class. Again, the 1-class model is used for initial start values and each model is estimated 100 times using the '`gridsearch()`' function. The summary of GMM-1 results looks like this:

```
     G     loglik npm      AIC      BIC %class1 %class2 %class3
gmm1 1 -1041.9742   4 2091.948 2102.369     100
gmm2 2  -855.3405   8 1726.681 1747.522      50      50
gmm3 3  -838.7191  12 1701.438 1732.700      35      45      20
```

Here, we can see that compared to the LCGA models, additional free parameters are estimated per model (one additional parameter per class). These are the class-specific random intercept variances. We can also see that the BIC values are considerably lower than for the LCGAs, indicating that the GMMs describe the data considerably better. Next, a specific model can be inspected more closely, for instance, by typing `summary(gmm2)`.

```
Goodness-of-fit statistics:
     maximum log-likelihood: -855.34
     AIC: 1726.68
     BIC: 1747.52

Maximum Likelihood Estimates:

Fixed effects in the class-membership model:
(the class of reference is the last class)

                     coef      Se   Wald p-value
intercept class1  0.00813 0.20634  0.039 0.96856

Fixed effects in the longitudinal model:

                     coef      Se   Wald p-value
intercept class1  9.95919 0.14039 70.939 0.00000
intercept class2 10.49674 0.14642 71.687 0.00000
time class1       0.26815 0.04801  5.585 0.00000
time class2       1.76959 0.04843 36.539 0.00000


Variance-covariance matrix of the random-effects:
          intercept
intercept    0.3874

                                  coef      Se
Proportional coefficient class1  0.88937 0.22389
Residual standard error:         1.06407 0.03767
```

The output is very similar to the LCGA output, with the only difference being the added estimation of a class-specific random intercept variance, displayed in the '`Variance-covariance matrix of the random-effects`'. In `lcmm`, the random effect variance is estimated for the highest numbered class (here: class 2) and the other class-specific variances are derived by multiplying the estimated (co)variance matrix by a proportional coefficient, which is provided at the bottom of the output (i.e. '`Proportional coefficient class1`'. In the example output, we obtain an intercept variance of 0.88 for class 2 and an intercept variance of (0.3874*0.89)=0.34 for class 1.

3.1.2 GMM-2 with lcmm

To run a GMM with both random intercepts and slopes, the GMM-1 code needs only be slightly adjusted as follows:

```
set.seed(2002)
gmm1_2 <- hlme(y ~ time, subject = "ID", random=~1 + time, ng = 1,
            data =mydata)
gmm2_2 <- gridsearch(rep = 100, maxiter = 10, minit = gmm1_2,
            hlme(y ~ time, subject = "ID", random=~1 + time,
                 ng = 2, data = mydata, mixture = ~ time, nwg=T))
gmm3_2 <- gridsearch(rep = 100, maxiter = 10, minit = gmm1_2,
            hlme(y ~ time, subject = "ID", random=~1+time,
                             ng = 3, data = mydata, mixture = ~ time,
                             nwg=T))

# make table with results for the 4 models:
summarytable(gmm1_2, gmm2_2, gmm3_2)
```

Here, the term '`random=~1+time`' indicates that we want to estimate a random intercept and slope ('`random=~time`' would accomplish the same; the intercept is included by default). After running the model with our simulated data, the summarized results look like this:

```
       G    loglik npm      AIC      BIC %class1 %class2 %class3
gmm1_2 1 -857.5434   6 1727.087 1742.718     100
gmm2_2 2 -828.7719  10 1677.544 1703.595      50      50
gmm3_2 3 -825.9719  14 1679.944 1716.416      18      50      32
```

We can see that each model has two more freely estimated parameters than the GMM with random intercept because a random slope and the random intercept-slope covariance are estimated in each class in this model configuration. Again, it can be seen that the model fit of these models is better than that of the less complex versions (LCGA and GMM-1). In addition, we can see that the 2-class model – unsurprisingly given the used simulation model - best describes the data, as indicated by the lowest AIC and BIC values (also see **Table 2**). In addition, the class-numbers remain the same compared to LCGA and GMM-1. The model output for this 2-class model can be further inspected (‘`summary(gmm2_2)`’) and looks as follows:

```
Goodness-of-fit statistics:
     maximum log-likelihood: -828.77
     AIC: 1677.54
     BIC: 1703.6

Maximum Likelihood Estimates:

Fixed effects in the class-membership model:
(the class of reference is the last class)

                     coef      Se   Wald p-value
intercept class1  0.00521 0.20572  0.025 0.97980

Fixed effects in the longitudinal model:

                     coef      Se   Wald p-value
intercept class1 10.49948 0.12268 85.586 0.00000
intercept class2  9.95285 0.11975 83.113 0.00000
time class1       1.76169 0.06725 26.195 0.00000
time class2       0.26605 0.06194  4.295 0.00002


Variance-covariance matrix of the random-effects:
          intercept    time
intercept   0.18818
time       -0.05754 0.10088

                                    coef      Se
Proportional coefficient class1  1.11805 0.22247
Residual standard error:         0.92713 0.03775
```

Here, the ‘`Variance-covariance matrix of the random-effects`’ now has three entries: the intercept variance, slope variance and intercept-slope covariance for the highest numbered class (class 2). The corresponding values for class 1 are obtained by multiplying the class-2 matrix by the provided proportional coefficient.
Here, the use of a proportional coefficient to estimate all remaining class-specific (co)variance matrices has the advantage that it requires estimation of less parameters (only a proportional coefficient per added class) than in an arrangement where the (co)variance matrix has to be estimated for each class separately (which is the approach of *Mplus* and *flexmix*). This becomes clear when comparing the numbers of free parameters of the GMM-1 and GMM-2 models between *lcmm* and the other packages (see **Table 2).** As a result of this approach, the estimated models in *lcmm* are more parsimonious. In addition, computations are more efficient reducing the time needed to estimate the models. However, these advantages do come at the expense of the rather strong assumption that the all elements in each class-specific random effects matrix can indeed be adequately approximated by multiplication of the estimated matrix with a single proportional coefficient.

3.2 *Package flexmix*

3.2.1 *LCGA in flexmix*

To run LCGA models with increasing numbers of classes in the *‘flexmix’* package, we can run the ‘`LXMRglmfix()`’ function within the ‘`stepFlexmix()`’ function. The latter allows for automatically stepping through LCGA models with increasing numbers of classes. The code looks as follows:

```
# open package:
library(flexmix)

# set random number seed:
set.seed(0111)

# estimate lcga models with 1-4 classes:
lcgaMix <- stepFlexmix(.~ .|ID, k = 1:4,nrep = 50, model =
FLXMRglmfix(y~ time, varFix=TRUE), data = mydata, control =
list(iter.max = 500, minprior = 0))

# inspect and compare model-fit results:
lcgaMix

# extract a specific model object (2-class model) from the
# lcgaMix object:
lcga2 <- getModel(lcgaMix, which=2)

# inspect the overall model and parameters of the 2-class
# model:
summary(lcga2)
parameters(lcga2)

# calculate class sizes by dividing the provided sizes by
# the number of time points per subject, for example:
lcgaMix@models$`2`@size/5
```

The `stepFlexmix()` function starts by defining the model formula. Here, a default `.~.` is entered, which will is updated with the formula provided in the `model` statement. The added element `‘|ID’` indicates that measures are nested within subjects. The `k=1:3` statement indicates that we want to estimate models with 1 to 3 classes, and `‘rep=50’` indicates that we want to run each model 50 times (to prevent local maxima). The `model` statement is used to specify the growth model. Here, we make use of the function `FLXMRglmfix()`, which allows users to specify general linear models for mixture analyses, while fixing certain parameters to be equal across classes. Here, we specify that we want the residual term to be equal across classes by `varFix=TRUE` This leads to estimation of a LCGA with a single residual term that reflects the overall unexplained variance in the data and does not vary across classes. This is analogous to the default in *Mplus* and *lcmm*. If no such constraints are required, `FLXMRglm()` can be used instead of `FLXMRglmfix()`. The final statement, `control= list(iter.max = 500, minprior = 0)` is used to (re)set relevant model estimation parameters.
When run, this code yields an object that contains information on all estimated models. Her, we named this object `lcgaMix`, which contains the following results:

```
  iter converged k k0     logLik      AIC      BIC      ICL
1    2      TRUE 1  1 -1146.3234 2298.647 2311.291 2311.291
2    6      TRUE 2  2  -875.4560 1762.912 1788.200 1788.269
3   10      TRUE 3  3  -858.2982 1734.596 1772.528 1784.414
```

Here, we can see the fit statistics for LCGA models with 1-3 classes. Using the `getModel()` function we then can take a closer look at the 2-class model. First we get the model summary `(summary(lcga2))`, showing the individual class proportions, log likelihood value, degrees of freedom and the BIC:

```
       prior size post>0 ratio
Comp.1   0.5  250    265 0.943
Comp.2   0.5  250    255 0.980

'log Lik.' -875.456 (df=6)
AIC: 1762.912   BIC: 1788.2
```

Interestingly, the AIC values found with *flexmix* correspond to those found using other software, whereas the BIC values are different, suggesting that the latter are calculated differently in *flexmix*. Indeed, as with the class size calculations, the package also uses the total number of data points (subjects*repeated measures) as sample size (n) in the calculation of the BIC (BIC=ln(n)k - 2ln($\hat{L}$)), whereas the other covered packages use the number of subjects[5]. Note that in line with the BIC calculation, the presented class-sizes reflect the number of data points per class, rather than the number of subjects. The number of subjects is easily calculated by dividing the size-column (`` lcgaMix@models$`2`@size ``) by the number of repeated measurements (here 5).

To look at the class-specific model-parameters, we can type (`parameters(lcga2)`), which in this case yields the following results:

```
                     Comp.1    Comp.2
coef.(Intercept) 9.9558053 10.498145
coef.time        0.2656535  1.766542
sigma            1.2149648  1.214965
```

Here, we see the class-specific intercepts and slopes (i.e. `coef.time`) and the residual term (`sigma`), which is the same across classes, as requested in the code. Unsurprisingly, the overall LCGA results are very similar to those found using the other software packages (see **Tables 1** and **3**).

3.2.2 GMM-1 in flexmix

To run a GMM with a random intercept using the *flexmix* package, we use the ‘`LXMRlmm()`’ function instead of ‘`LXMRglmfix()`’ function in ‘`stepFlexmix()`’:

```
# set random number seed
set.seed(1234)

# estimate gmm models with 1-4 classes
gmmMix <- stepFlexmix(.~ .|ID, k = 1:4, nrep = 100, model =
FLXMRlmm(y~ time, random = ~ 1, varFix = c(Random = FALSE,
Residual = TRUE)), data = gmmdata, control = list(iter.max =
1000, minprior = 0))

# inspect and compare model-fit results
gmm_Mix

# extract a specific model object from the gmmMix object.
gmm2 <- getModel(gmmMix, which =2)

# inspect the model, for example:
summary(gmm2)
parameters(gmm2)

# calculate class sizes by dividing the provided sizes by the
# number of time points per subject, for example:
gmmMix@models$`2`@size/5
```

The general structure of the code is similar to that for the LCGA, except for the used function ‘`FLXMRlmm()`’, which enables to use a linear mixed models function, and thus, inclusion of both fixed and random effects. The code for the GMM includes the term ‘`random=~1`’ to indicate that we want to estimate a random intercept[6]. The element `varFix=c(Random=FALSE, Residual=TRUE)` is used to indicate if the random effect variance and residual variance should be fixed across classes or estimated for each class separately. Here, the intercept variances are estimated class-specific by stating `Random=FALSE` and the residual variance is kept constant across classes by stating `Residual=TRUE`. After running this code, the model output looks like this:

[5] For instance, for the 2-class LCGA model, the BIC calculation used in lcmm is: ln(100)*6 – 2(-875.46)=1778.5 and the calculation used in flexmix is: ln(500)*6 – 2(-875.46)=1788.2.

```
  iter converged k k0     logLik      AIC      BIC      ICL
1    8      TRUE 1  1 -1041.9742 2091.948 2108.807 2108.807
2   24      TRUE 2  2  -855.3408 1726.682 1760.399 1760.901
3   34      TRUE 3  3  -838.7213 1701.443 1752.018 1763.516
```

The AIC and BIC are notably lower than those for the LCGA models, indicating that these models generally provide a better description of the data. If we select the 2-class model for further inspection, we get the following information:

```
        prior size post>0 ratio
Comp.1 0.498  250    285 0.877
Comp.2 0.502  250    275 0.909

'log Lik.' -855.3408 (df=8)
AIC: 1726.682   BIC: 1760.399
```

We can see that the class sizes are the same as those found in the LCGA. The model parameter estimates can be inspected next (‘`parameters(gmm2)`’):

```
coef.(Intercept) 10.4950739 9.9597174
coef.time         1.7696437 0.2682005
sigma2.Random     0.3883217 0.3084847
sigma2.Residual   1.1317975 1.1317975
```

The shown coefficients include the fixed intercept (‘`coef.(Intercept)`’)and slope (`coef.time`) for each of the estimated classes. The requested class-specific intercept variances for a are provided in the row ‘`sigma2.Random`’. Finally, the overall residual variance is provided (‘`sigma2.Residual`’) and fixed across classes, as requested in the code.

3.2.3 *GMM-2 with flexmix*

To run a GMM with random intercepts and slopes, the code needs only to be slightly adjusted, by adding time to the random effects to be estimated (`random=~1+time`):

```
# set random number seed
set.seed(1234)

#Run models with 1-4 classes
gmmMix2 <- stepFlexmix(.~ .|ID, k = 1:3, nrep = 100, model
= FLXMRlmm(y~ time, random = ~ 1 + time,
varFix=c(Random=F, Residual=T)), data = mydata,
control=list(iter.max = 1000, minprior = 0))

gmm2_2 <- getModel(gmmMix2, which =2)
```

After running the code, we can again take a look at the summary of the model-fit indices in the model output:

```
  iter converged k k0    logLik      AIC      BIC      ICL
1   32      TRUE 1  1 -857.5481 1727.096 1752.384 1752.384
2  611      TRUE 2  2 -828.3619 1680.724 1731.299 1732.411
3  827      TRUE 3  3 -824.5532 1685.106 1760.969 1787.593
```

The models took much more iterations to converge compared to LCGA and GMM-1, but all estimations did eventually reach convergence, as shown in the second column (‘`converged`’). When comparing the AIC and BIC, we can see that the lowest values are obtained for the 2-class model, indicating – not surprisingly – that this model best describes the simulated data. Further inspection of this two-class model (i.e. ‘`summary(gmm2_2)`’) shows us:

```
       prior size post>0 ratio
Comp.1 0.501  250    330 0.758
Comp.2 0.499  250    310 0.806

'log Lik.' -828.3619 (df=12)
AIC: 1680.724   BIC: 1731.299
```

Here, we can see that both estimated classes are of equal size, similar to the LCGA and GMM-1 results. Finally, we can take a look at the estimated model parameters (i.e. ‘`parameters(gmm2_2)`’), which look like this:

```
                        Comp.1      Comp.2
coef.(Intercept) 10.49933978  9.95371893
coef.time         1.76293406  0.26677424
sigma2.Random1    0.28890857  0.09849563
sigma2.Random2   -0.07222707 -0.04435989
sigma2.Random3   -0.07222707 -0.04435989
sigma2.Random4    0.11462830  0.10657086
sigma2.Residual   0.86561728  0.86561728
```

The intercepts (‘`coef.(Intercept)`’) and slopes (‘`coef.time`’) as well as the residual variance (‘`sigma2.residual`’) are quite easily identified and similar in magnitude as in the LCGA and GMM-1 results. The other parameters represent the estimated class-specific elements of the intercept and slope (co)variance matrix, which has the following structure.

	Intercept	Slope
Intercept	`sigma2.Random1`	`sigma2.Random2`
Slope	`sigma2.Random3`	`sigma2.Random4`

The first random term (‘`sigma2.Random1`’) represents the class-specific intercept variance. The entries (‘`sigma2.Random2`’ and ‘`sigma2.Random3`’, represent the off-diagonal elements of the matrix (i.e. the random intercept-slope covariances). These both have the same value as the matrix is symmetric within each class. Finally, the term ‘`sigma2.random4`’ represents the class-specific slope variance. The parameters are presented in a more comprehensible way in **Table 3**.

3.3 Comparison between R-packages and Mplus

3.3.1 Comparison of modeling results

As there are plenty of tutorials on the use of Mplus for LCGA and GMM (see e.g., www.statmodel.com), this section will not go into the code and output, but will merely use the results of the demonstrated LCGA, GMM-1 and GMM-2 analyses when conducted in *Mplus*. These results will be used for comparison to the R-packages.

The fit results of the analyses are shown in **Table 1** and **2**. The model fit of the LCGA and GMMs is mostly comparable to that found with the R-packages. The number of estimated parameters per GMM model in *Mplus* is higher than in *lcmm* and the same as in *flexmix*, as the latter and *Mplus* both estimate a complete random effects covariance matrix per class. All packages showed the lowest BIC for the 2-class GMM-2 model, and thus identified the same model as the optimal choice.

The 2-class model parameter estimates of *Mplus* and the R-packages are shown in **Table 3**. Here, we can again see that the obtained results do not differ strongly, with some slight variations in the (non-significant) covariance estimates. A stable estimate of the intercept-slope covariance was not expected, as the data were simulated to contain zero covariance between slopes and intercepts.

Table 3: estimated coefficients for the 2-class models estimated by LCGA and GMM in simulated data

			Software package		
Model	Class	parameter	*lcmm*	*flexmix*	*Mplus*
LCGA	*Class1*	*Intercept*	*10.5*	*10.0*	*10.5*
		Slope	*1.8*	*0.3*	*1.8*
	Class1	*Intercept*	*10.0*	*10.5*	*10.0*
		Slope	*0.3*	*1.8*	*0.3*
GMM-1	*Class1*	*Intercept*	*10.0*	*10.5*	*10.0*
		Intercept variance	*0.3*	*0.4*	*0.3*
		Slope	*0.3*	*0.3*	*0.3*
	Class1	*Intercept*	*10.5*	*10.0*	*10.5*
		Intercept variance	*0.4*	*0.3*	*0.4*
		Slope	*1.8*	*1.8*	*1.8*
GMM-2	*Class1*	*Intercept*	*10.5*	*10.5*	*10.0*
		Intercept variance	*0.2*	*0.3*	*0.1*
		Slope	*1.8*	*1.8*	*0.3*
		Slope variance	*0.1*	*0.1*	*0.1*
		Slope-intercept covariance	*-0.06*	*-0.07*	*-0.05*
	Class 2	*Intercept*	*10.0*	*10.0*	*10.5*
		Intercept variance	*0.2*	*0.1*	*0.3*
		Slope	*0.3*	*0.3*	*1.8*
		Slope variance	*0.1*	*0.1*	*0.1*
		Slope-intercept covariance	*-0.06*	*-0.04*	*-0.08*

3.3.2 Comparison of modeling approaches

Looking at the tutorials and the results obtained with the different packages based on the same data, we can see that the same general results are obtained with each of the covered packages and that the R-packages yield results that are similar to those found in *Mplus*. However, each package takes a slightly different approach to the default configuration of the models to be estimated and the calculation of the indices of fit, which can lead to different results. Some notable differences are listed below:

- The model residuals can be estimated in different ways when doing LCGA or GMM and this is reflected in the packages' default settings. In *lcmm*, a single residual is estimated for the complete model. The default in *flexmix* (FLXMRglmfix and FLXMRlmm) is to estimate residuals for each class. In *Mplus*, the default is to estimate a single residual for each time point but not for each class separately. In each package, the estimation of the residuals can be adjusted to a user's need.
- The estimated log likelihood values are similar across packages, but the calculation of the indices of fit differed slightly. *Lcmm* and *Mplus* compute the BIC using the total number of subjects for n. *Flexmix* uses the total number of measurements (i.e. subjects*time points) for *n*, leading to comparatively higher BIC values. In either case, it is very easy to (re)calculate the BIC with a different *n* if this is required.
- The class-specific random effects co-variance matrices are estimated differently by *lcmm* compared to *flexmix* and *Mplus*. *Lcmm* estimates a single matrix for the highest numbered class and yields proportional coefficients that can be used to derive the matrices for the other classes. *Flexmix* and *Mplus* estimate the full matrix for each class. As a result, computation times are generally longer for the latter packages.

4. *Final remarks*

The current tutorial presented relatively straightforward LCGA and GMM examples. In many cases, additional steps or model elements may be required, such as the fitting of non-linear growth functions or the use of alternative link functions (e.g., count or binary outcomes). In addition, many real datasets will contain missing values that need to be dealt with in some way. These topics were outside the scope of the current tutorial, but will be briefly discussed below.

4.1 Missing data

In theory, LCGA and GMM have the advantage that they can be estimated using full information maximum likelihood (FIML). This means that, assuming that the missing data are missing at random (MAR; Collins et al., 2001), both subjects with complete and with incomplete data can be included in the analysis and the model will be estimated based on the available data. This approach has been fully implemented in Mplus, but not – to our knowledge – in the covered R-packages. This means that missing data have to be handled using an alternative approach. Here, the optimal choice would be to use multiple imputation (e.g., Schafer & Graham, 2002), e.g., using the package ‘mice’ (van Buuren, & Groothuis-Oudshoorn, 2011). In terms of implementation, this would require more steps and be considerably more time consuming than FIML, but results would be equivalent.

4.2 Covariates

It is possible to add covariates to LCGA and GMM models. Such variables can be added, adjusting for (1) class-membership probabilities (e.g., higher level on baseline severity score makes one more likely to be in class x) and/or (2) for variations in the intercept or slope within the class (e.g., a higher score on a severity s ale makes one more probable to have a higher intercept). In *lcmm*, covariates with both fixed or random effects in the growth model can easily be added as part of the model formula arguments in `hlme()`. In addition, the `classmb` argument can be used to add covariates of class membership. In *flexmix*, covariates can be included with fixed or random effects as well, but tere is no dedicated argument that can be used to add covariates of class-membership.

4.3 Non-linear growth

The demonstrated examples can be easily extended to mixture models based on non-linear growth. If such growth is plausible, the growth model that underlies the LCGA or GMM can be extended with a non-linear term (e.g., quadratic or cubic), or can be changed to e.g., a logarithmic function. Addition of such terms is easily done in all covered packages.

4.4 Other R packages

For the current tutorial, only two R-packages were selected. However, before selecting *lcmm* and *flexmix* more packages were reviewed. Eventually, the latter two were selected mainly because they contain functions that make it relatively straightforward to write the code for estimating latent classes based on linear mixed growth models. This makes them relatively user-friendly and accessible for applied researchers. However, other R-packages exist that can be used to conduct the same kinds of analyses (and much more). ‘*OpenMX*’ (Neale et al., 2016) is an excellent all round SEM package that can also be used to run GMM. The package ‘*mixAK*’ (Komárek & Komárková, 2014) allows for (Bayesian) estimation of mixtures based on generalized linear mixed models and can also be used for GMM and related analyses, including those with multivariate outcome data. The package ‘mixtools’ (Benaglia et al., 2009) is a general package for finite mixture modeling, with the added option to estimate latent classes in non-parametric settings and based on regression models with various link functions. Each of these packages

References

Benaglia T, Chauveau D, Hunter DR, Young D (2009). “mixtools: An R Package for Analyzing Finite Mixture Models.” Journal of Statistical Software, 32(6), 1–29.

Collins, L. M., Schafer, JL. et al. (2001). A comparison of inclusive and restrictive strategies in modern missing data procedures. Psychological Methods 6(4): 330-351.

Curran PJ, Obeidat K, Losardo D. (2010). Twelve Frequently Asked Questions About Growth Curve Modeling. J Cogn Dev.11(2):121-136.

Leisch F (2004). FlexMix: A general framework for finite mixture models and latent class regression in R. Journal of Statistical Software, 11(8), 1-18.

Jones, B. L., Nagin, D. S., & Roeder, K. (2001). A SAS procedure based on mixture models for estimating developmental trajectories. Sociological Methods & Research, 29, 374–393

Jung T. Wickrama, KAS (2008). An Introduction to Latent Class Growth Analysis and Growth Mixture Modeling. Social and Personality Psychology Compass 2/1: 302–317.

Komárek A, Komárková L (2014). Capabilities of R Package mixAK for Clustering Based on Multivariate Continuous and Discrete Longitudinal Data. Journal of Statistical Software, 59(12)

Lazarsfeld PF (1950). The interpretation and computation of some latent structures. In S. A. Stouffer et al., Measurement and prediction. Princeton: Princeton Univ. Press, Ch. 11.

Lazarsfeld PF, Henry P (1968). Latent structure analysis. New York : Houghton Mifflin.

Mislevy RJ, Verhelst N (1990). Modeling item responses when different subjects employ different solution strategies. Psychometrika 55: 195–215

Muthén, L. K., & Muthén, B. O. (2013). Mplus: Statistical analysis with latent variables. User’s guide. Los Angeles, CA: Muthén & Muthén

Neale MC, Hunter MD, Pritikin JN, Zahery M, Brick TR, Kirkpatrick RM, Estabrook R, Bates TC, Maes HH, Boker SM. OpenMx 2.0: Extended Structural Equation and Statistical Modeling. Psychometrika. 2016 Jun;81(2):535-49.

Proust-Lima C, Philipps V, Liquet B. Estimation of extended mixed models using latent classes and latent processes: the R package lcmm. J Stat Softw 2017;78:1–56

Schafer JL, Graham, J (2002) Missing Data: Our View of the State of the Art. Psychological Methods 7: 147-177.

Rabe-Hesketh, S., Skrondal, A., & Pickles, A. (2004). GLLAMM manual. U.C. Berkeley Division of Biostatistics Working Paper Series. Working Paper 160.

Ram, N., & Grimm, K. J. (2009). Methods and measures: Growth mixture modeling. A method for identifying differences in longitudinal change among unobserved groups. International Journal of Behavioral Development, 33, 565–576

van Buuren, S., Groothuis-Oudshoorn, K. (2011). MICE: Multivariate Imputation by Chained Equations in R. Journal of Statistical Software, 45(3), 1–67.

Vermunt, JK, Magidson J. (2016). Technical guide for latent GOLD 5.1: Basic, advanced, and syntax. Belmont, CA: Statistical Innovations.

Appendix 1: Simulation code

```
#Download and open package MASS
library(MASS)

#Function to simulate longitudinal data with random intercept(s) and
slope(s)
ml_data <- function(n_pers, n_time, beta_int, beta_slo_time,
beta_slo_covar, beta_slo_interact, mean_i, var_i, mean_s, var_s, cov_is,
mean_r, var_r) {

  Rand.eff = mvrnorm(n_pers, mu=c(mean_i,mean_s), Sigma=rbind(c(var_i,
cov_is),
                                                        c(cov_is, var_s) ))

  colnames(Rand.eff) = c("intercept","slope_time");

  dat= data.frame(ID   = rep(1:n_pers,   each=n_time),
                  time = rep(1:n_time,   times=n_pers),
                  covar= c(rep(0,n_pers/2, each=n_time),rep(1,n_pers/2,
each=n_time)),
                  int = rep(Rand.eff[,1],  each=n_time),
                  slo = rep(Rand.eff[,2], each=n_time),
                  slo_cov=rep(beta_slo_covar, each=n_time),
                  slo_interact=rep(beta_slo_interact, each=n_time),
                  y = NA)

  dat$time <- dat$time-1
  dat$interact <- dat$time*dat$covar

  y = with(dat, (beta_int + int) + (beta_slo_covar)*covar
           + (beta_slo_time + slo)*time
           + (beta_slo_interact)*interact
           + rnorm(n=n_pers*n_time, mean=mean_r, sd=sqrt(var_r)))
  dat$y <-y

  return(dat) }

# simulate the tutorial data
set.seed(2002)
d1 <- ml_data(n_pers=100,
              n_time=5,
              beta_int=0,
              beta_slo_time=0.3,
              beta_slo_covar=0.5,
              beta_slo_interact=1.5,
              mean_i=10,
              var_i=0.13,
              mean_s=0,
              var_s=0.09,
              cov_is=0,
              mean_r=0,
              var_r=1)

# trim the number of variables
mydata <- d1[,c(1,2,3,8)]
```

Appendix 2: used Mplus code

All code shown for the 2-class models. Note that each model was run with 1, 2 and 3 classes (by changing the `classes=c()` statement and by adding/removing the relevant class-specific parameters (i.e. `%c#1%, %c#2% and %c#3%`)

LCGA

```
Data: file="mydata_wide.dat";

Variable:
! provide column names and indicate which
! variables to include in analyses

names are rownr ID c y1 y2 y3 y4 y5;
usevariables are y1 y2 y3 y4 y5;

! State how many classes to estimate
classes=c(2);

! provide analysis details
Analysis:
type=mixture;
starts=100 20;
estimator=MLR;

! Define the overall growth model and
! determine which parameters are estimated !
overall and which are estimated for each ! class
specifically.
Model:
%overall%
! define growth model
i s | y1@0 y2@1 y3@2 y4@3 y5@4;
! fix intercept and slope variances to
! zero:
i@0;
s@0;
! constrain residuals to be equal across
! time points:
y1-y5 (1);

! For each of the classes request the mean (fixed)
intercept and slope to be estimated.
%c#1%
[i-s];

%c#2%
[i-s];
```

GMM-1

```
Model:
%overall%
i s | y1@0 y2@1 y3@2 y4@3 y5@4;
s@0;
y1-y5 (1);

%c#1%
[i-s];
i;

%c#2%
[i-s];
i;
```

GMM-2

```
Model:
%overall%
i s | y1@0 y2@1 y3@2 y4@3 y5@4;
y1-y5 (1);

%c#1%
[i-s];
i-s;
i with s;
```

```
%c#2%
[i-s];
i-s;
I with s;
```