Comparing methods of classifying life courses: Sequence Analysis and Latent Class Analysis

Sapphire Y. Han Aart C. Liefbroer Cees H. Elzinga

Appendix 1: Practical considerations

In the main body of this article, substantive and methodological aspects of SA and LCA were discussed. Here, we will focus on some practical issues that arise from applying these models. Two aspects of the sequence data set – the size of the data set and sequence length – affect the choice of typology techniques, the computation duration and the computation capacity requirements. In this Appendix, we will cover the effect of each of these two aspects on SA and LCA typologies.

To conduct SA, a data set of size N (the total number of respondents) will generate a distance matrix of size $N \times N$ and therefore computation time and memory requirement will increase quadratic with N. For big data sets, the memory requirements may easily exceed the capacity of a standard PC. For LCA, the memory requirements are only proportional to N and a bigger data set will facilitate the convergence of the estimation algorithm. Therefore, bigger data sets will not normally pose a computational problem in terms of memory requirements or computation time for an LCA.

Sequence length, i.e. the total number of states contained in a sequence, may be a problem, both for SA and for LCA.

For SA it may pose a computational problem since sequence comparison, i.e. the computation of distances, requires computation time that increases at least quadratic with sequence length. That means that doubling the length will increase computation time required by a factor of four. So, if, like in OM, the states are repeated to express duration, a 15-year long sequence with monthly observations will consist of 180 states. Such pairs of sequences require quite some processing time to count the minimum number of edits required (for details of the OM- algorithm, the reader is referred to Brzinsky-Fay et al. (2006)). However, some distance metrics like SVRspell (Elzinga & Studer, 2015) and OMspell

(Studer & Ritschard, 2016), treat the durations as properties of the states to the effect that the sequence length is substantially reduced, mostly by a factor of more than 10.

Long sequences also pose a computational problem to LCA as, over time, each observation is a realisation of a separate independent variable of which the class-conditional probability distribution over the distinct states has to be estimated. With R latent classes, J distinct states and K observations per sequence, i.e. a sequence length of K, the number of parameters to be estimated is in the order of $R \times J \times K$ (see also Equation 1). Therefore, models with many latent classes applied to long sequences may require infeasible computational capacity.

Another issue that may seriously affect the computational duration and the result of applying LCA, is the issue of convergence and local maxima of the log-likelihood function. The surface of the log-likelihood function L to be maximised when estimating the parameters of the LC-model, is irregular and, depending on the parameters and the data, has many local maxima. The EM-algorithm (Dempster et al., 1977) used to iteratively maximise L will always find a maximum on that surface but it might be very slow in closely approximating it (Wu, 1983). Furthermore, such a maximum may well be a local maximum that is quite smaller than the true, global maximum and there is no reliable method that avoids convergence to a local maximum. This can be problematic because in a local maximum, the parameter-estimates might be quite different from those that would have been found, had the global maximum been hit. EM works iteratively and it has to be provided with an initial guess of the parameters. At each iteration cycle, these guesses are updated until the increase of L is negligible and this may well occur in the neighborhood of a local maximum. One way to try to circumvent this problem is to run the algorithm many times (in our study, 1000 runs per model were performed) with different initial guesses of the parameters and search for an ever-bigger value of L. In practice, such a random search will generate an increase of L. There are alternatives to such a random search (see e.g. Watson & Engle, 1983) but none of them is guaranteed to find the global maximum of L. Using the estimates that comprise a local maximum may imply that some of the sequences will be assigned to a latent class that differs from the one that would have been assigned with a better local maximum or the global maximum. Therefore, the results of an LCA will

always suffer, to an unknown degree, of such uncertainties. Unfortunately, in practice, using many initial values to better approximate the global maximum of L may be extremely time-consuming. We encountered this problem when trying to find an optimum for the number of latent classes. LCA-software such as LatentGOLD (Vermunt & Magidson, 2005) and poLCA (Linzer & Lewis, 2011) automatically generates and tests different random start values, and therefore running the program many times can reduce the risk of ending up with a local maximum. Note that it appears unwise to assume that the results of a single run of any LCA-program will produce a global maximum, even with a small number of latent classes.

Appendix 2: A heuristic bridge between SA and LCA

As concisely explained in the section Latent Class Analysis, we propose an interpretation of the LC-model that implies that variation of the ordering of states is almost absent under the LC-model. Furthermore, we propose one extra assumption that bridges the difference between the probabilistic LC-model and the distance-oriented SA-approach. Thereto, we formally develop an idea on the stochastic interpretation of edit-distances that was suggested in Kruskal (1983, pp. 232-234). Similar ideas in the context of automatic data-editing were proposed by Liepins (1980) and by Scholtus (2014) and in the field of speech recognition by Bahl and Jelinek (1975) and Ristad and Yianilos (1998). Below, we will develop Kruskal's sketchy proposal to relate the OM-distance to the likelihood of generating one sequence from another through a stochastic mutation process and use this development to relate SA- and LCA-based typologies of sequences. The Latent Class model and the Sequence Analysis approach are quite different. First, the LC-model is about unobserved probability distributions that characterise different groups of sequences while no such distributions play any role in the SA-approach. Furthermore, the LC-model expresses the probability of observing particular, *individual* sequences while in the SA-approach, only *pairs of sequences* and their distances are considered to construct a grouping of the observations. Distances have no place in the LC-model. So, connecting models as different as the SA-approach and the LC-model requires a few steps and assumptions.

The first step in this process is to provide for an interpretation of the LC-model that involves an editing process and a template-sequence. Once this interpretation is established, the

second step involves simplifying the model until we arrive at a direct connection between observation probabilities and the distances to the template sequence.
Finally, in the third step we use a general property of distance metrics - the triangular inequality - to establish a simple relation between edit distances between pairs of sequences and the probability that such pairs have been edited from the same template, i.e. the probability that they stem from the same class.

An edit-process interpretation of the LC-model

Here, we revisit the LC-model in order to arrive at a more convenient interpretation of it. The LC-model states that individuals, in the form of encoded life courses or patterns of responses to test-items, each belong to distinct groups that are characterised by distinct probability distributions over some alphabet of states or responses. Let us denote the $(d+1)$-sized alphabet as a set $A = \{a_0, a_1, \cdots, a_d\}$ wherein, for reasons to become clear, the symbol a_0 is considered as ''empty''. All observed sequences are constructed from this alphabet but do not necessarily contain all symbols available. So, given an n-long sequence $x = x_1 \cdots x_n$, we know that the subsequent states have been taken from A. Under the LC-model, we hypothesize a set $\Theta = \{\theta_1, \cdots, \theta_k\}$ of k latent classes and each sequence is supposed to be generated from precisely one of these classes. Furthermore, the LC-model assumes that, given the sequence is generated from a particular class θ_ℓ, the consecutive symbols of the sequence are generated independently - the assumption of ``local independence''. Thus, under the LC-model, the probability of observing $x = x_1 \cdots x_n$, given that x comes from class θ_ℓ, can be written as

$$Prob(x|\theta_\ell) = \prod_{i=1}^{n} Prob(x_i|\theta_\ell) \qquad (A2.1)$$

As all symbols x_i come from the alphabet A, the class θ_ℓ is characterised by the probability distribution $Prob(A|\theta_\ell) = \{Prob(a_0|\theta_\ell), \cdots, Prob(a_d|\theta_\ell)\}$. Put more abstractly, a class is characterised by its relative size $Prob(\theta_\ell)$ and the sampling process through which it generates subsequent symbols from the alphabet. The sampling process is with replacement, has no memory and is governed by the distribution $Prob(A|\theta_\ell)$. Estimating an LC-model thus amounts to estimating the class-sizes $Prob(\theta_\ell)$ and the conditional sampling distributions $Prob(A|\theta_\ell)$.

Let us now develop an interpretation of the LC-model that will allow us to connect observation probabilities to edit-distances. To do just that, we assign to each class θ_ℓ its own model- or template sequence $\tau_\ell = \tau_{\ell 1}, \cdots, \tau_{\ell n}$ and an ``editor'' E_ℓ that generates observable sequences by editing the unobservable, latent template τ_ℓ. The template might look like

$$aaaaaabbbbccccccccbba,$$

while the edited version might look like

$$aaaaabbbbbbccbbccaaa.$$

We assume that this editor samples edit-operations and applies the operation sampled to the symbol encountered when processing τ_ℓ. The operations are limited to either deleting a symbol from τ_ℓ or inserting or substituting a symbol from the alphabet A. Thus, the editor is characterised by a matrix

$$\boldsymbol{E_\ell} = \begin{pmatrix} p_{00} & \cdots & p_{0d} \\ \vdots & \ddots & \vdots \\ p_{d0} & \cdots & p_{dd} \end{pmatrix}$$

wherein p_{ij} denotes the probability that, when the editor encounters the symbol a_i, it will change it into the symbol a_j:

$$p_{ij} = Prob\left(x_m = a_j \middle| \tau_{\ell m} = a_i, \boldsymbol{E}_\ell\right).$$

Now the empty symbol a_0 is convenient since if $i = 0 \neq j$, the edit is an insertion of a_j; if $i \neq 0 = j$, the edit is a deletion and if $i \neq 0 \neq j$, the edit is a substitution and if $i = j$, nothing changes.

Clearly, so far, we did not add any assumption to the LC-model that has consequences for the probabilities, given class-membership, of the observed sequences. All we did was postulate the existence of editors that operate on different templates as an explanatory mechanism that generates the observed sequences. We did not touch the basic LC-model assumption that is embodied in the above Equation (A2.1).

Simplifying the model

When an LC-model fits the data well, we may expect that within classes, the observed sequences are quite similar. In our interpretation of the model, we would say that we expect that most observed sequences are quite similar to the templates from which they were

generated. Hence we may expect that change is relatively rare. The reader notes that this is an interpretation that follows from assuming that the LC-model fits the data quite well: if the LC-model would not show a reasonable fit, it would not make much sense to study the connection between the LC-model and any other approach. Stated in terms of the entries of the matrix of edit-probabilities, this implies that all diagonal elements p_{ii} are close to 1 and that all off-diagonal entries are relatively small. A good approximation of this interpretation is to say that all p_{ii} are equal to some number p close to 1 and that all entries p_{ij} are equal to some number q that is relatively close to 0. So, we must have that

$$\boldsymbol{E}_{\ell}^{*} = \begin{pmatrix} p & q & \cdots & q \\ q & p & \cdots & q \\ \vdots & \vdots & \ddots & \vdots \\ q & q & \cdots & p \end{pmatrix} \approx \boldsymbol{E}_{\boldsymbol{\ell}}$$

i.e. if the sequences within classes are quite homogeneous, the $\boldsymbol{E}_{\boldsymbol{\ell}}$ are well approximated by the simpler matrices $\boldsymbol{E}_{\boldsymbol{\ell}}^{*}$.

The second assumption about the editing process is that the editor has no memory, i.e. at each position of the template, the editor samples an edit-operation according to the matrix $\boldsymbol{E}_{\boldsymbol{\ell}}^{*}$. So, we assume that consecutive edits are independent of each other. The reader notes that this is an assumption that follows from the LC-model assumption of local independence; it is not an *additional* assumption that is chosen independently of the LC-model.

When the editor processes the template τ_{ℓ}, it creates an edit-path: a sequence of changes and non-changes to the symbols of the template. Assuming that the individual edit-probabilities are specified by $\boldsymbol{E}_{\boldsymbol{\ell}}^{*}$ and that the edits are independent, the probability of the whole path can be written as a multiplication of p's and q's: a p for each position where nothing changes and a q for all position were something is changed. If there are n_p positions where no change is applied and $n - n_p = n_q$ positions where some edit is applied, this multiplication, i.e. the probability that this edit path is realised, can be written as

$$p^{n_p} \cdot q^{n_q}.$$

Of course, given some template τ_{ℓ} and some observable sequence x, there will be many different edit paths that will turn the template into the observable sequence. Let us denote

the set of all edit paths that turn τ_ℓ into x by $\mathbb{E}_\ell\ (x) = \{e_1, e_2, \cdots\}$ and consider an arbitrary path e_i from this set. Then there will be $n_p(i)$ locations where no edit will be applied in this path and $n_q(i)$ positions where some change will be affected. Hence the probability of the occurrence of this particular path will be

$$p^{n_p(i)} \cdot\ q^{n_q(i)}.$$

Given the template τ_ℓ, the probability that x will be generated depends on the probability of all paths in the set $\mathbb{E}_\ell\ (x)$:

$$Prob(x|\tau_\ell) = \sum_{e_i \in \mathbb{E}_\ell\ (x)} Prob(e_i)$$

$$= \sum_{e_i \in \mathbb{E}_\ell\ (x)} p^{n_p(i)} \cdot\ q^{n_q(i)}$$

$$\approx \sum_{e_i \in \mathbb{E}_\ell\ (x)} q^{n_q(i)}$$

The last approximation is justified by the ``fact'' that p is close to 1, hence $p^{n_p(i)}$ must be close to 1 too.

The reader notes that the bigger $n_q(i)$, the smaller $q^{n_q(i)}$ since q itself is a small number. Hence the summands in the above approximation are small and the bigger the exponent, i.e. the longer the edit path, the smaller the contribution to the sum. This explains our assumption that there is a shortest edit path, say e^* consisting of only n^* edits, of which the probability dominates the last sum. If this assumption is correct, we must have that

$$Prob(x|\tau_\ell) \approx \sum_{e_i \in \mathbb{E}_\ell\ (x)} q^{n_q(i)} \approx q^{n^*} \qquad (A2.2)$$

This latter approximation is the key result of this subsection as it explicitly relates the probability of observing some sequence x to the length of the shortest edit path that is required to generate the observed sequence from the template sequence τ_ℓ. This assumption is *not* a consequence of our interpretation of the LC-model, but follows from our

attempt to connect a local-independence model to a distance metric that is based on some minimum (weighted) number of edits. The reader also notes that the simplification of the $\boldsymbol{E}_\ell$-matrix to $\boldsymbol{E}_\ell^*$ is not essential: it only considerably simplifies our notation. What is essential though, is the assumption that the diagonal elements are big, i.e. close to 1, and that the off-diagonal elements are very small, i.e. close to 0.
The last step to be made is to connect this probability to an edit distance; the subject of the next subsection.

From probabilities to edit-distance
Our last challenge is to remove all references to edit-probabilities from the right-hand side of the last of the above approximations. This is accomplished by assigning a weight $w = -\log q$ to each edit in the path, implying that $q = e^{-w}$. Substituting this weighting into the last approximation yields

$$Prob(x|\tau_\ell) \approx e^{-wn^*} = e^{-wd(x,\tau_\ell)}$$

wherein $d(x, \tau_\ell)$ denotes the OM-distance between the template and the observed sequence. This is the key-result, applied in the present context, of reasoning along the lines set out by Kruskal (1983): the likelihood of observing x given the template is a decreasing function of the distance between x and the template. The bigger the distance, the less likely it is that x was generated from that template.

Unfortunately, as the template cannot be observed, this result is not directly usable. However, the purpose of the analysis of the observed sequences, either through LCA or through SA, is to group sequences into more or less homogeneous classes such that sequences end up in the same group if they are similar. So we are interested in evaluating the relation between $Prob(x, y|\tau_\ell)$ i.e. the likelihood that x and y were generated from the same template or belong to the same latent class, and the distances $d(x, y)$. So, we need to make one more step.

Since the editor is assumed to have no memory, i.e. because of the local independence assumption of the LC-model, the observables x and y are generated independently. Hence

$$Prob(x, y|\tau_\ell) = Prob(x|\tau_\ell) \cdot Prob(y|\tau_\ell)$$

$$\approx e^{-w\left(d(x,\tau_\ell)+d(y,\tau_\ell)\right)}$$

$$\leq e^{-wd(x,y)}$$

The last step derives from the fact that d is a metric and thus satisfies the triangular inequality $d(x,y) \leq\ d(x,z) + d(z,y)$ for all triples x, y and z (see e.g. Elzinga & Studer, 2014). Setting $z = \tau_\ell$ then yields the last inequality.

This last inequality is our final result: the bigger the unit-cost edit-distance, the smaller (the upper bound of) the probability that the observables x and y stem from the same template, i.e. from the same latent class. This is a nice result as it implies that the grouping obtained through partitioning a distance matrix (SA) should roughly coincide with grouping that is obtained through LCA.

This result, as presented here, is obtained through a simplified argument: it uses the simpler $\boldsymbol{E}_\ell^*$ instead of the full $\boldsymbol{E}_\ell$. This simplification forced us to consider unit-cost OM. However, had we chosen to use the full, non-simplified $\boldsymbol{E}_\ell$, then we would have obtained a similar result for an arbitrary but metric cost-matrix. We do not present this extended argument here because of a lack of space and because it does not really add to our understanding of the heuristics. Of course, if in practice the structure of the edit-cost matrix is inappropriately chosen not to be metric, reasoning via the triangular inequality fails. Furthermore, the reader should be aware that this result is obtained through the assumption that there is one and only one shortest path whose likelihood dominates the sum of likelihoods of all possible paths. If this assumption is false, i.e. if there are many edit-paths that are suboptimal but almost as likely as the shortest path, the approximation is an underestimation of the true likelihood. Therefore, we investigate this assumption in some more detail in the next subsection.

On the number of shortest edit paths or longest common subsequences

In this subsection, we will make some remarks on the maximum number of shortest edit paths. To do so, we will be concerned with the dual problem of the number of longest common subsequences (lcs's). We know that the OM-distance with unit-cost, the so-called

Levenshtein distance (Levenshtein, 1966), counts the number of symbols in either sequence that do not belong to an lcs of both sequences: the smallest number of edits required to turn the one sequence into a perfect copy of the other sequence. So, if there are many distinct shortest and short edit-paths, there must be many distinct lcs's and the reverse is also true: many distinct lcs's implies many distinct editing paths. We know that (Greenberg, 2003) often the number of lcs's is quite big and we also know that (see Elzinga, 2014b) the maximum number of distinct lcs's that may exist for sequences of a given length increases exponentially with the length of the sequences. The maximum number $f(n,k)$ of k-long lcs's of two n-long sequences is given by

$$f(n,k) = \prod_{i=0}^{k-1} \left\lfloor \frac{n+i}{k} \right\rfloor$$

The latter equation generates a triangular table, a small part of which is shown in Table 1.

Table 1: Part of the triangular table for the number $f(n,k)$ of k-long lcs's in pairs of n-long sequences for $10 < n \leq 15$ and $1 \leq k \leq n$.

n/k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
11	11	30	48	54	48	32	16	8	4	2	1				
12	12	36	64	81	72	64	32	16	8	4	2	1			
13	13	42	80	108	108	96	64	32	16	8	4	2	1		
14	14	49	100	144	162	144	128	64	32	16	8	4	2	1	
15	15	56	125	192	243	216	192	128	64	32	16	8	4	2	1

Table 1 shows the values of $f(n,k)$ for $10 < n \leq 15$. For example, it is shown that the maximum number of 7-long lcs's for two sequences of length $n = 14$ equals 128: $f(14,7) = 128$. The reader might verify that $f(20,7) = 1458$. What do these numbers imply? That multiple lcs's are common among pairs of social science sequences? We don't think so for a number of reasons. First, we note that the numbers in Table 1 are *maxima*; pairs of random sequences will, on average, have much less lcs's. Second, within classes or clusters, pairs of sequences are quite similar and similar sequences will have relatively long lcs's which are much less abundant than medium sized lcs's (see Table 1). Finally, in most kinds of social science sequences, we observe two phenomena: changes of state are

relatively rare as is variation in the *order* of states. These two phenomena imply that, especially within the same class or cluster, social science sequences are structured like

$$aa \ldots\ldots abb \ldots\ldots bcc \ldots\ldots cdd \ldots\ldots :$$

Thus, they consist of only a few long runs of the same state and variation mainly consists in variation in the lengths of consecutive runs. It is difficult to see how pairs of such sequences from the same class or cluster could have even a few distinct lcs's. Rather, such pairs will have many *distinct embeddings* of the same lcs. Therefore, while acknowledging that the above reasoning does not constitute a proof of the validity of the assumption of just one shortest, dominant edit path (Equation A2.2), we believe that it shows that this assumption is very plausible.

So, from the above heuristics, we conclude that there is a conceptual bridge between SA and the LC-model: the idea that distinct latent classes can be envisaged as arising from distinct template sequences that are stochastically perturbed and that, therefore, one may expect that results obtained with either approach will roughly coincide. Mathematically, these heuristics only rely, beyond LCA's assumption of local independence, on the assumption of a single, shortest edit path and numerically on the assumption that the p_{ii} of $\boldsymbol{E}_{\ell}$ are "sufficiently" close to 1. If these assumptions fail, the heuristics lack relevance. Whatever the case, at present, we do not know of a more plausible narrative to explain the apparent correspondence between results obtained with LCA and results obtained with SA.

Appendix 3: Code for Sequence Analysis with R-based tools

```
############################### Section 1: sequence analysis
#############################

################################ install necessary packages
##############################

# TraMineR is a R-package for describing, summarizing, analyzing and rendering discrete
sequence #data

install.packages(TraMineR)

require(TraMineR)  # load the TraMineR package for sequence analysis

# The WeightedCluster library provides functions to cluster states sequences and weighted
data. install.packages(WeightedCluster)

require(WeightedCluster) # load the WeightedCluster package for cluster quality analysis

# fpc package provides bootstrapping methods and statistics for clustering.

install.packages(fpc)

require(fpc) # load the fpc package for bootstrapping

################################# set working directory
#################################

setwd ("replace this with your working directory")

################################### load sequence data
#################################

load("data_monthly_sequence.RData") # replace RData with your own sequence data

#################### here we start preparing the analysis with TraMineR
###################

data.lab <- c("Married", "Married with Children", "Single", "Single with Children",
```

```
"Union", "Union with Children") # 6 demographic events

data.alph <- c("M","MC","S","SC","U","UC") # define the sequence alphabet with these 6 events

data.shortlab <- c("M","MC","S","SC","U","UC") # abbreviation of these 6 demographic events

data.seq <- seqdef(data_monthly_sequence, states = data.shortlab, labels = data.lab, alphabet = data.alph) # creates a state sequence object with attributes such as alphabet and state labels

########################## calculate sequence distance matrix c4.dis ######################

############################## using optimal matching (OM) ############################

####### insertion and deletion cost equals 4, substitution-cost is constant (default value = 2) ######

data.dis <- seqdist(data.seq, indel = 4,method = "OM",sm = "CONSTANT")

############################## other option see help file seqdist ########################

############## here we apply hierarchical cluster analysis using Ward's method ##############

hc.ward <- hclust(as.dist(data.dis), method = "ward.D")

######### here we compare cluster quality between  2 and 8 clusters ############

mvad <- wcKMedRange(data.dis, kvals = 2:8,initialclust = hc.ward)

######### here we apply bootstrapping for 7 clusters as an example ############

data.hc <- clusterboot(data.dis, distances = TRUE, clustermethod = disthclustCBI, method = "ward.D", k = 7)
```

here we perform data visualization for the cluster solution to gain substantive understanding

###################### We take 7 clusters as an example ######################

#################### and require sequence index plots ####################

seqIplot(data.seq, group = data.hc$partition, border = NA, weighted = FALSE, sortv = "from.start")

################### sequence medoid plot for cluster number equals 7 ###################

################### calculate sequence medoid for cluster number equals 7 #################

icenter <- disscenter(data.dis, factor(data.hc$partition), medoids.index="first")

seqiplot(data.seq[icenter,]) ## plot calculated medoid for the 7-cluster solution

########################## Section 2: latent class analysis ############################

################################ install necessary packages ##############################

poLCA is a R package for latent class analysis and latent class regression models for polytomous #outcome variables. Also known as Latent Structure Analysis.

install.packages(poLCA)

require(poLCA) # load the poLCA package for latent class analysis

################# we start preparing the latent class analysis using poLCA ##################

#################### We use the same dataset as in sequence analysis ##################

```
##################### rename the monthly sequence states chronologically
#################

names(data_monthly_sequence)[8:151] <- paste("m", 1:144, sep = "")

##################### define variables in latent class analysis ###################

var <- cbind(m1, m2, m3, m4, m5, m6, m7, m8, m9, m10,
        m11, m12, m13, m14, m15, m16, m17, m18, m19, m20,
        m21, m22, m23, m24, m25, m26, m27, m28, m29, m30,
        m31, m32, m33, m34, m35, m36, m37, m38, m39, m40,
        m41, m42, m43, m44, m45, m46, m47, m48, m49, m50,
        m51, m52, m53, m54, m55, m56, m57, m58, m59, m60,
        m61, m62, m63, m64, m65, m66, m67, m68, m69, m70,
        m71, m72, m73, m74, m75, m76, m77, m78, m79, m80,
        m81, m82, m83, m84, m85, m86, m87, m88, m89, m90,
        m91, m92, m93, m94, m95, m96, m97, m98, m99, m100,
        m101, m102, m103, m104, m105, m106, m107, m108, m109, m110,
        m111, m112, m113, m114, m115, m116, m117, m118, m119, m120,
        m121, m122, m123, m124, m125, m126, m127, m128, m129, m130,
        m131, m132, m133, m134, m135, m136, m137, m138, m139, m140,
        m141, m142, m143, m144)~1

##################### here we perform latent class analysis ###################

##################### with 7 latent classes as an example ###################

## the number of random starting values  nrep may be changed
```

```
m7 <- poLCA(var, data_monthly_sequence, nclass = 7, nrep = 500)

### Getting model fit statistic BIC and relative entropy for latent between 4 and 8.

##################### BIC calculated in a for loop ###################

BIC <- NULL

for (i in 4: 8)

{

 m <- poLCA(var, data_monthly_sequence, nclass = i, nrep = 500)

 BIC[i] <- m$bic

}

##################### relative entropy calculated in a for loop ###################

entropy <- NULL

for (i in 4: 8)

{

 m <- poLCA(var, data_sts, nclass = i, nrep = 100)

 entropy[i] <- poLCA.entropy(m)

}

# here we perform data visualization for the classification solution to gain substantive understanding

###### data visualization of latent class analysis classification using TraMineR package #########

####################### and use the 7-cluster solution as an example ######################

class=m7$predclass
```

```
##################### sequence index plot for class number equals 7
#####################

seqIplot(data.seq, group class, border = NA, weighted = FALSE, sortv = "from.start")

##################### sequence model plot for cluster number equals 7
####################

seqmsplot(data.seq, group = class, border = NA, weighted = FALSE)



########################## Section 3: Typology Comparison
############################

################################## install necessary packages
##############################

# Software for multinomial log-linear models.

install.packages(nnet)

require(nnet) # load the nnet package for multinomial logistic regression analysis

##################### load data of cluster and classification solution
#####################

##################### load data of background variable ####################

load("background.Rda")

##################### load sequence analysis results ####################

load("Sequence_solution.RData")

##################### load latent class analysis results ####################

load("Latent_class_solution.RData")

##################### factorize all categorical variable ####################
```

```
f_Sequence_solution <- factor(Sequence_solution)

f_ Latent_class_solution <- factor(Latent_class_solution)

f_background_country <- factor(background$country)

##################### using Netherland as reference country ####################

f_background_country <- relevel(f_background_country, ref = "13")

##################### using traditional as reference cluster ####################

f_Sequence_solution <- relevel f_Sequence_solution, ref = "3")

##################### using traditional as reference cluster ####################

############## perform multinomial logistic regression on cluster solution ##########

glm.fit.sa=multinom(country  ~ f_Sequence_solution, data= f_Sequence_solution)

################################### calculate BIC
#####################################

BIC(glm.fit.sa)

########################## Predict probability ###################################

dses <- data.frame(c("1", "2", "3", "4", "5", "6")) ###create a 6 –cluster dataset for
sequence analysis

names(dses) <- "sequence_solution"

predict(glm.fit.sa, newdata = dses, "probs")
```

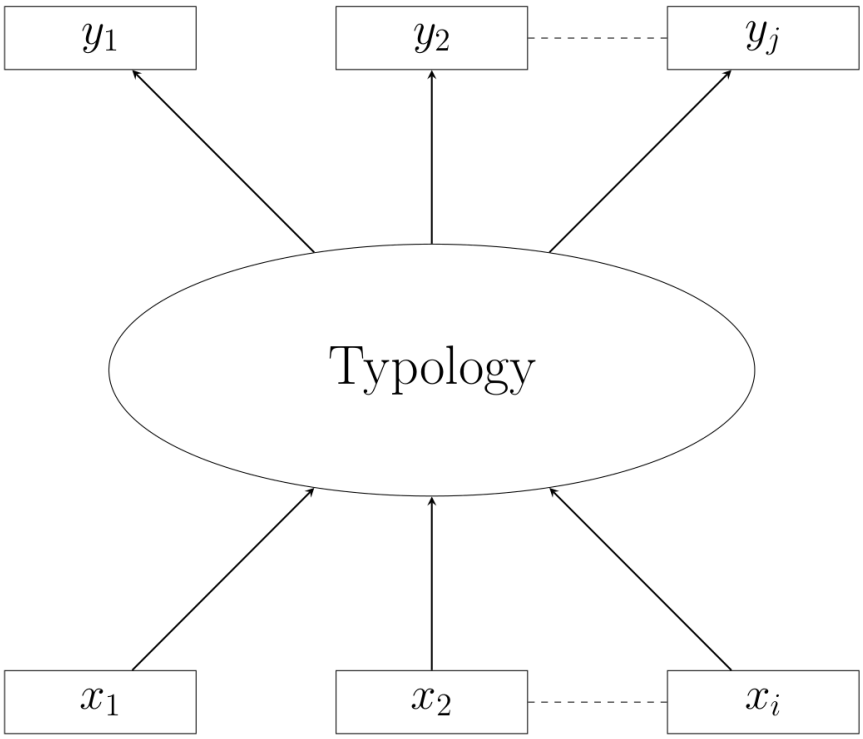
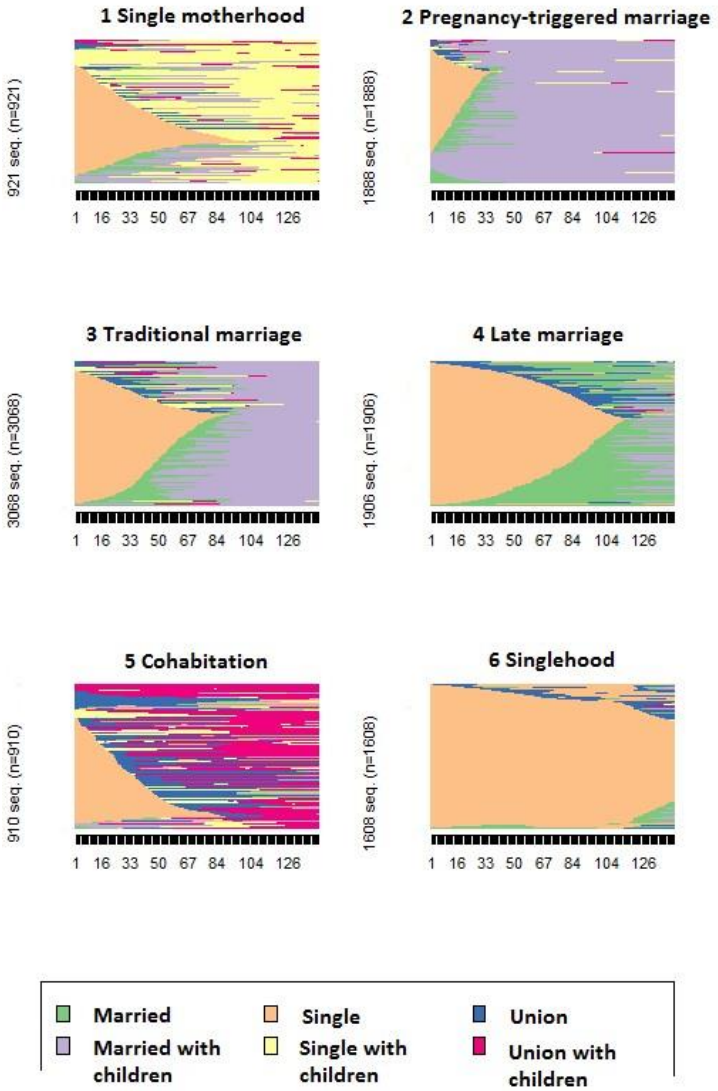
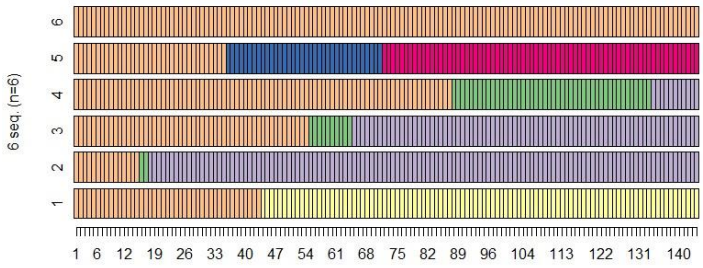


Figure 1. A representation of a nomological network surrounding a life course typology

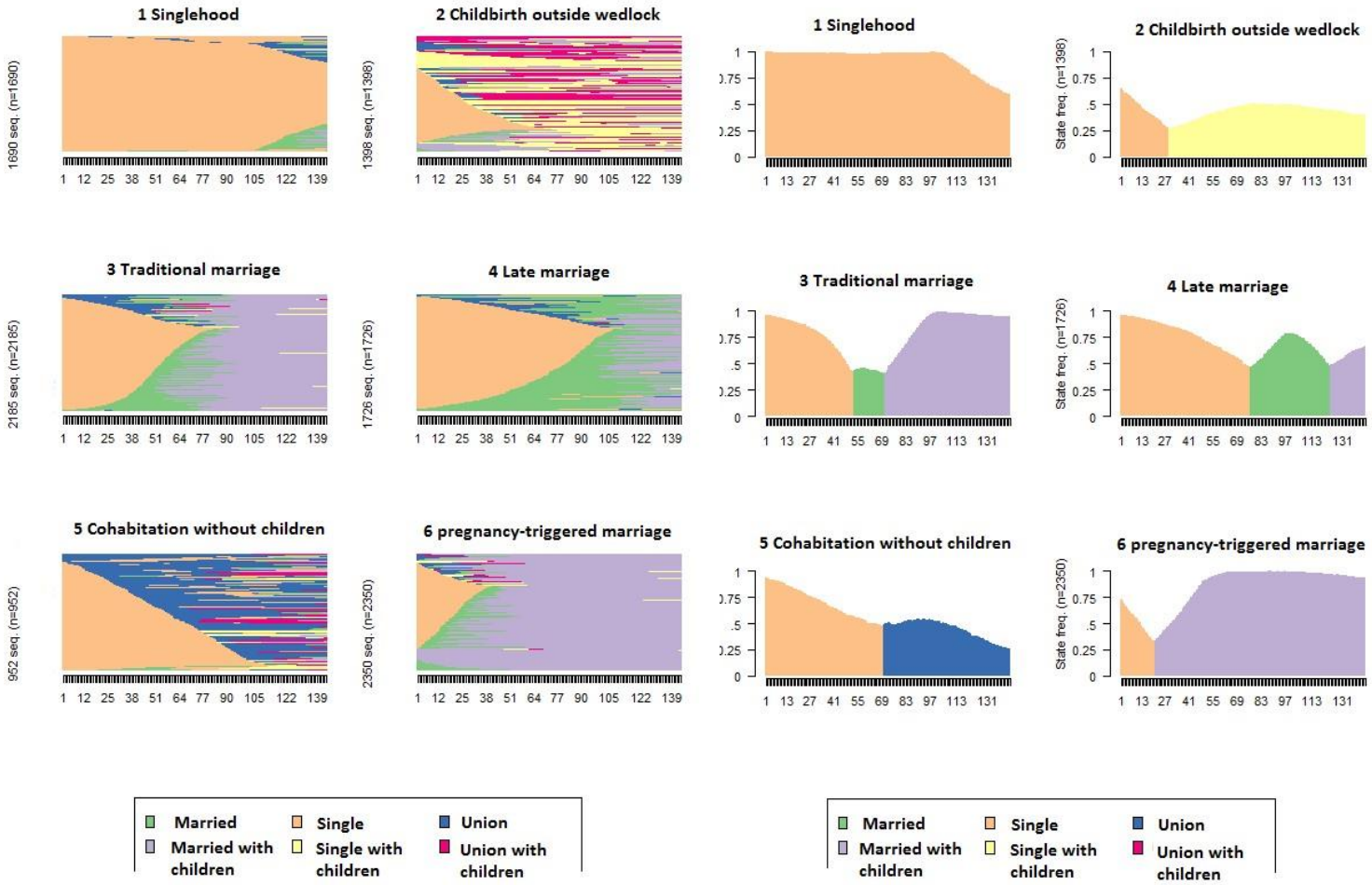


(a)



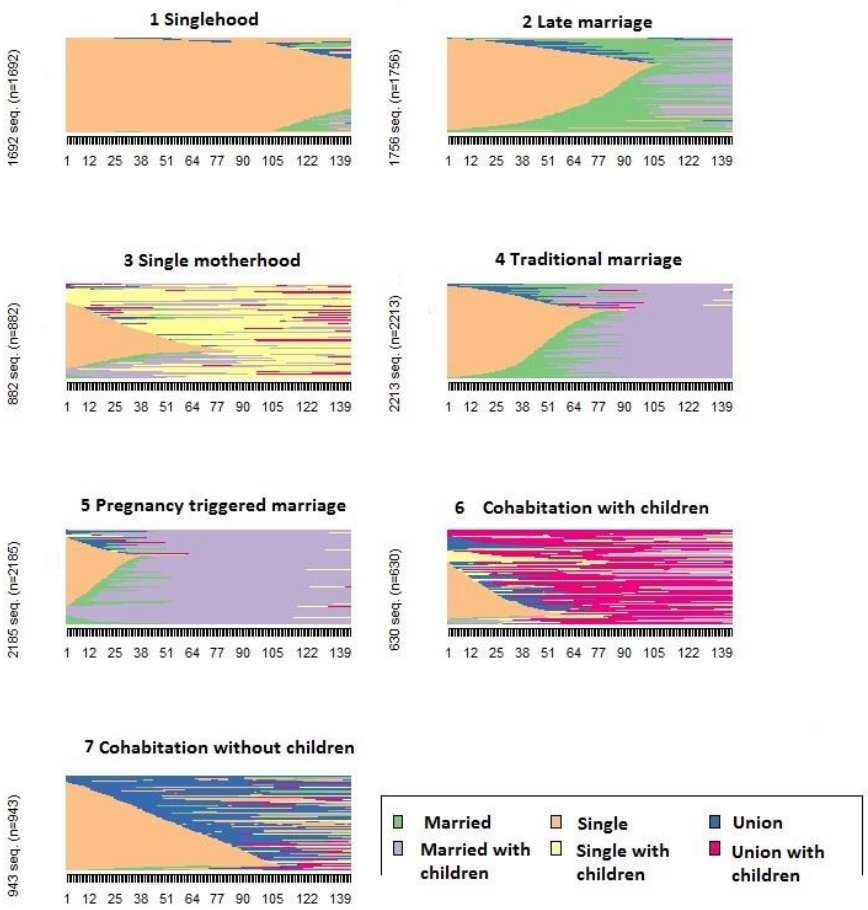
(b)

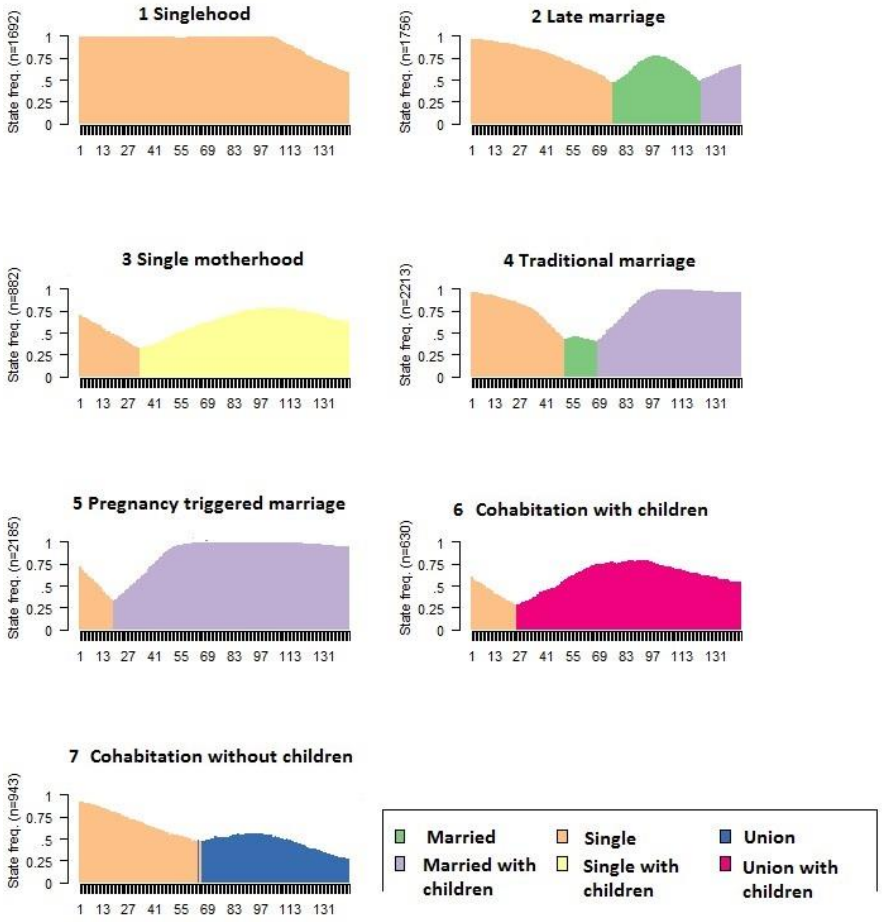
Figure 2. Sequence Index plot (a) and Sequence Medoid plot (b) of the OM-6 solution



(a) (b)

Figure 3. Sequence Index plot (a) and Sequence Model State plot (b) of the LCA-6 solution


1 Singlehood
1692 seq. (n=1692)
1 12 25 38 51 64 77 90 105 122 139
2 Late marriage
1756 seq. (n=1756)
1 12 25 38 51 64 77 90 105 122 139
3 Single motherhood
882 seq. (n=882)
1 12 25 38 51 64 77 90 105 122 139
4 Traditional marriage
2213 seq. (n=2213)
1 12 25 38 51 64 77 90 105 122 139
5 Pregnancy triggered marriage
2185 seq. (n=2185)
1 12 25 38 51 64 77 90 105 122 139
6 Cohabitation with children
630 seq. (n=630)
1 12 25 38 51 64 77 90 105 122 139
7 Cohabitation without children
943 seq. (n=943)
1 12 25 38 51 64 77 90 105 122 139
Married
Single
Union
Married with children
Single with children
Union with children



(a) (b)

Figure 4. Sequence Index plot (a) and Sequence Model State plot (b) of the LCA-7 solution.

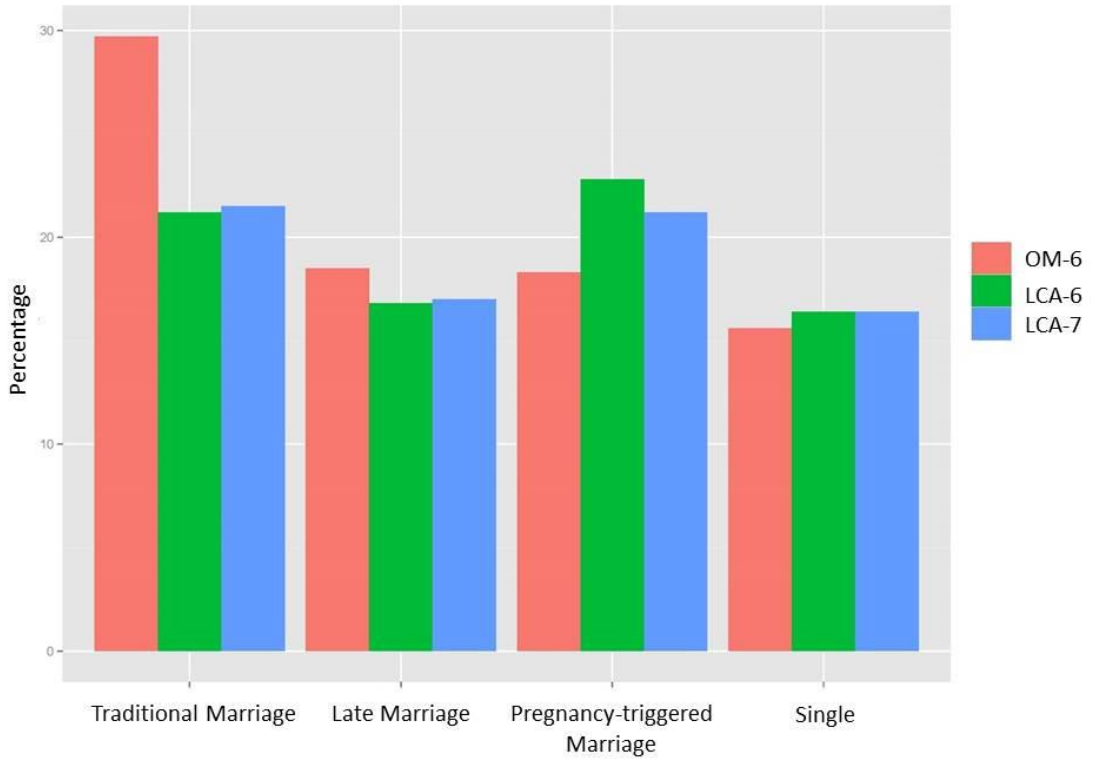
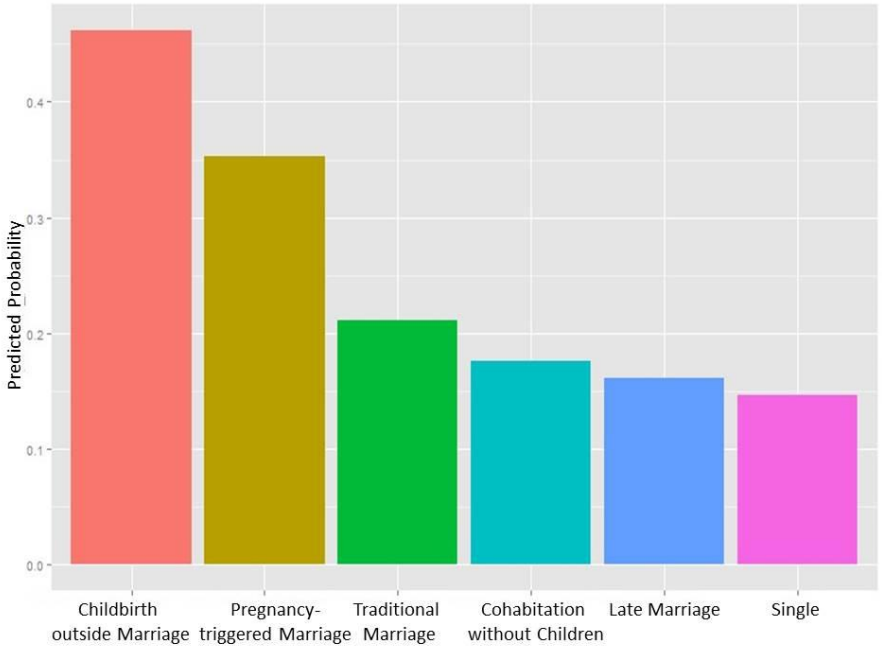
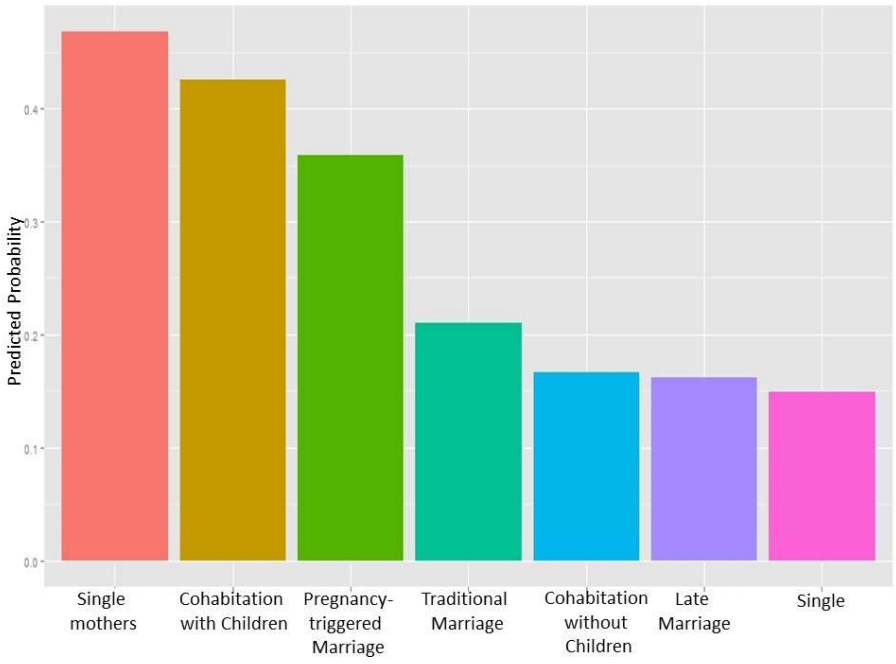


Figure 5. Percentage of respondents in the four large clusters or classes of OM-6, LCA-6 and LCA-7



(a)



(b)

Figure 6: Predicted probabilities of "no education after age 15" using a multinomial logistic regression model for (a) the LCA-6 solution and (b) the LCA-7 solution

Table 1. Definition of social background variables used in the typology comparison

Abbreviation	Meaning
Edu1	No education after age 15
Edu2	0-3 years of education after age 15
Edu3	3-5 years of education after age 215
Edu4	5+ years of education after age 15
Pardiv0	Parents not divorced
Pardiv1	Parents divorced
Pardiv3	Parents' divorce not known
Reli0	Not religious
Reli1	Catholic
Reli2	Protestant
Reli3	Other religion
Reli4	Religion unknown

Table 2. Number of respondents per country and percentage of the respondents per category of the social background variables

	Region	edu1	edu2	edu3	edu4	pardiv0	pardiv1	pardiv3	reli0	reli1	reli2	reli3	reli4	Nr. Resp.
1	Estonia	9.51	48.59	41.20	0.70	66.20	22.18	11.62	47.54	0.00	41.20	11.27	0.00	284
2	Czech Republic	5.10	23.81	50.34	20.75	85.37	14.29	0.34	--	--	--	--	--	294
3	France	8.02	44.32	22.94	24.72	87.53	10.91	1.56	--	--	--	--	--	449
4	New Zealand	--	--	--	--	--	--	--	--	--	--	--	--	460
5	Hungary	21.58	30.56	25.85	22.01	85.04	14.53	0.43	39.10	47.65	8.12	3.21	1.92	468
6	Latvia	0.85	22.67	39.41	37.08	77.12	19.07	3.81	31.78	20.55	17.37	26.69	3.60	472
7	Lithuania	1.17	15.37	33.46	50.00	80.54	18.09	1.36	8.17	80.93	0.78	8.56	1.56	514
8	Slovenia	18.02	20.14	32.69	29.15	92.40	7.42	0.18	21.02	68.90	0.18	8.66	1.24	566
9	Netherlands	4.08	35.85	22.84	37.22	85.02	9.98	4.99	39.94	34.80	19.21	5.90	0.15	661
10	Spain	36.95	22.36	10.84	29.85	97.32	2.68	0.00	17.40	78.05	0.54	3.08	0.94	747
11	Austria	20.88	30.05	33.51	15.56	90.03	9.44	0.53	30.72	59.18	3.59	6.38	0.13	752
12	Canada	--	--	--	--	82.72	15.31	1.96	3.80	46.73	33.77	15.71	0.00	764
13	Italy	30.47	15.48	18.43	35.63	97.79	2.21	0.00	8.72	90.05	0.49	0.61	0.12	814
14	Portugal	--	--	--	--	94.38	5.18	0.44	--	--	--	--	--	908
15	U.S.A.	50.65	0.28	8.47	40.60	75.84	24.07	0.09	8.47	29.05	49.67	12.71	0.09	2148

The table is ordered by the total number of respondents per region

" --" Data not available

Table 3. Values of cluster quality statistics

Number of clusters	PBC	ASW	HC
2	0.48	0.34	0.21
3	0.49	0.29	0.22
4	0.62	0.34	0.13
5	0.57	0.31	0.14
6	0.63	0.35	0.10
7	0.59	0.33	0.11
8	0.58	0.33	0.10

Note: PBC (maximal value preferred, ASW (maximal value preferred) and HC (minimal value preferred)

Table 4. Values of CJBM statistics of all six clusters of the OM optimal solution.

CJBM	Cluster 1	Cluster2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7
SA-6	0.57	0.66	0.54	0.49	0.45	0.76	NA
SA-7	0.62	0.69	0.54	0.42	0.48	0.46	0.76

Table 5. Values of latent class analysis model fit statistics: BIC (minimal value preferred, and relative entropy (closest to one value preferred)

Number of clusters	BIC*10^6	relative entropy
2	3.0	0.9993
3	2.6	0.9992
4	2.3	0.9982
5	2.1	0.9980
6	2.0	0.9979
7	1.9	0.9976
8	1.8	0.9975

Table 6: Cross tabulation of LCA and SA typology solutions, values shown as percentages.

LCA/SA	Cohab	Lmarriage	Pmarriage	Smothers	Single	Tmarriage
Cohab with c	4.14	0.00	0.04	0.45	0.00	1.50
Cohab without c	3.80	3.27	0.00	0.93	1.10	0.06
Lmarriage	0.12	13.03	0.00	0.11	0.17	3.62
Pmarriage	0.01	0.00	17.88	0.30	0.00	3.02
Smothers	0.75	0.08	0.32	6.46	0.00	0.96
Single	0.00	2.08	0.00	0.01	14.34	0.00
Tmarriage	0.03	0.05	0.09	0.69	0.00	20.63

Cohab = Cohabitation, Lmarriage = Late marriage, Pmarriage = pregnancy-triggered marriage, Smother = single mother, Tmarriage = Traditional marriage, Cohab with c = Cohabitation with children, and Cohab without c = Cohabitation without children

Table 7. BICs of SA (OM 6 and OM 7) and LCA (LCA 6 and LCA 7), based on multinomial logistic regression models

	Education	Parental divorce	Religion	Country
OM 6	21612.34	8970.52	20308.92	51758.54
OM 7	21626.74	8986.59	20333.26	51813.44
LCA 6	21448.98	8909.06	20225.49	51583.76
LCA 7	21441.70	8927.17	20230.11	51442.98